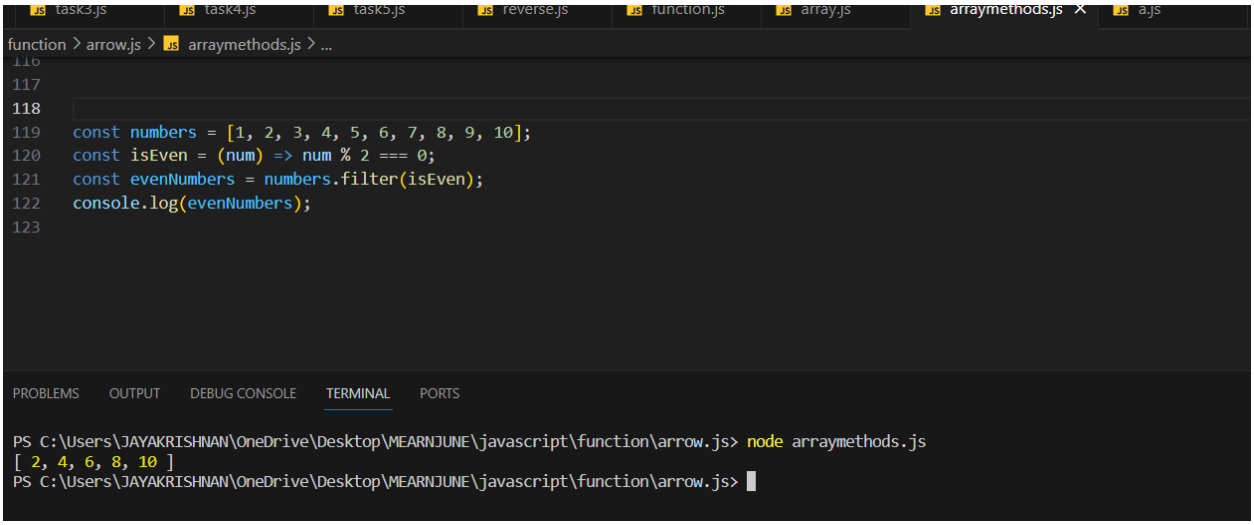
1. What is the purpose of the Array.prototype.filter() method in Javascript?
   Ans: The Array.prototype.filter() method in JavaScript is used to create a new array containing all elements of the calling array that pass a test (provided as a function). It does not modify the original array but returns a new array with the elements that satisfy the condition specified in the callback function.

2.How does the filter()method work can you explain the basic idea behind its functionality

Ans: he filter() method works by iterating through each element in an array and applying a callback function to each element. If the callback function returns true, the element is included in the new array; if it returns false, the element is excluded. Here's a step-by-step explanation of how it works:

- **Initialization**:
  The `filter()` method is called on an array.
  It takes a callback function as an argument.
- **Iteration**:
  The method iterates through each element of the array.
- **Callback Execution**:
  For each element, the callback function is executed with the current element, its index, and the entire array as arguments.
  The callback function returns a boolean value (`true` or `false`).
- **Condition Check**:
  If the callback returns `true`, the element is added to the new array.
  If the callback returns `false`, the element is skipped.
- **Result**:
  After all elements have been processed, `filter()` returns the new array containing only the elements that passed the test.

3.Can you demonstrate how to use the filter()method to create a new array of even numbers from an existing array of integers



4.How does the filter() method differ from find () method in terms of functionality and returned values

Ans: `filter()` Method

Functionality:

- The `filter()` method creates a new array with all elements that pass the test implemented by the provided callback function.
- It processes every element in the array and includes the elements for which the callback function returns `true` in the new array.

Returned Value:

- Returns a new array containing all the elements that pass the test.
- If no elements pass the test, it returns an empty array.

`find()` Method

Functionality:

- The `find()` method returns the value of the first element in the array that satisfies the provided testing function.
- It stops searching as soon as it finds the first element that passes the test.

Returned Value:

- Returns the value of the first element that satisfies the provided testing function.
- If no element passes the test, it returns `undefined`.

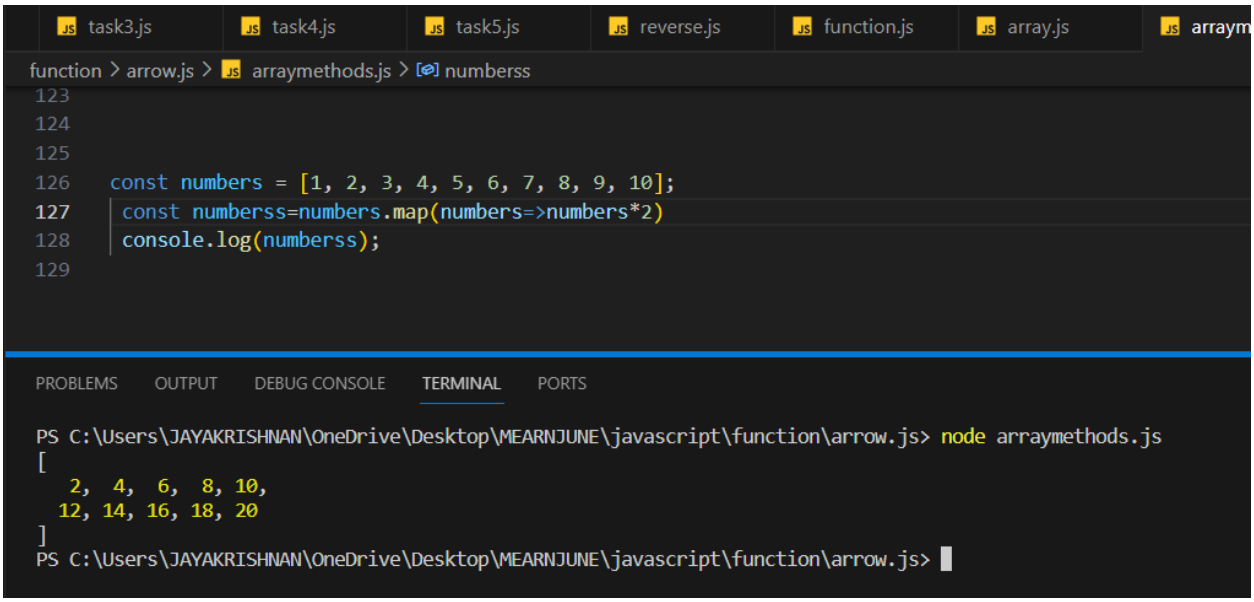
5. what is the purpose of array Array.prototype.map() method in javascript

Ans:The Array.prototype.map() method in JavaScript is used to create a new array by applying a provided function to each element in the original array. It is a powerful and commonly used method for transforming arrays.

Purpose:

The primary purpose of the map() method is to transform each element in an array based on a function that you provide. This function is called a callback function and is executed once for each element in the array, resulting in a new array containing the transformed elements.

6



.

7. can you explain the difference between the map()method and foreach() method

Ans: **Purpose**

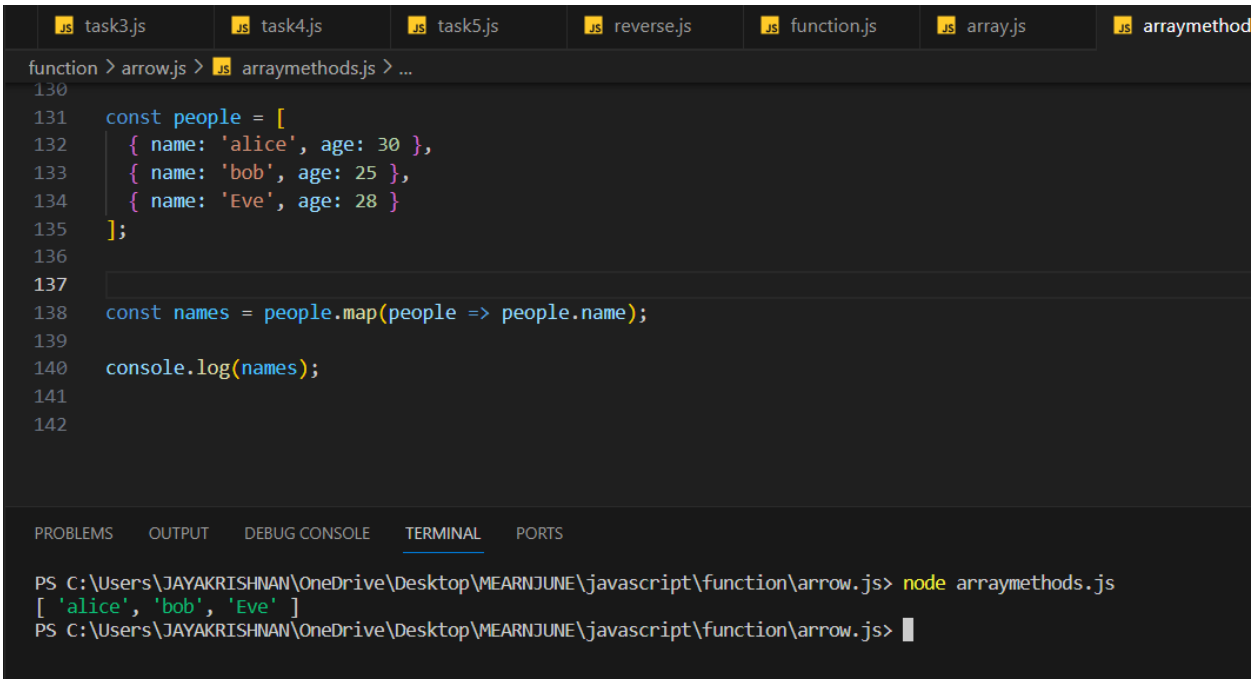
- **`map()`**: The `map()` method is used to transform each element in an array and create a new array with the transformed elements.
- **`forEach()`**: The `forEach()` method is used to execute a provided function once for each array element. It is typically used for performing side effects, such as logging or modifying elements in place.

Return Value

- **`map()`**: Returns a new array containing the results of applying the callback function to each element.
- **`forEach()`**: Returns `undefined`. It does not create a new array; it simply executes the provided function on each element of the array.

Mutability

- **`map()`**: Does not modify the original array. It creates and returns a new array with the transformed elements.
- **`forEach()`**: Can modify the original array if the callback function mutates the elements. However, `forEach()` itself does not return anything



9.How does the reduce() method work? Can you explain the basic idea behind its functionality

Ans: The basic idea behind its functionality is to iterate through an array and accumulate a result based on the elements in the array.

- **Initial Value and Accumulator**: The `reduce()` method starts with an initial value (if provided) and an accumulator. The accumulator is used to store the accumulated result from each iteration.
- **Callback Function**: The `reduce()` method takes a callback function as its first argument. This callback function is executed on each element of the array. The callback function takes four arguments:
- `accumulator`: The accumulated value previously returned in the last invocation of the callback (or the initial value, if provided).
- `currentValue`: The current element being processed in the array.
- `currentIndex`: The index of the current element being processed in the array.
- `array`: The array `reduce()` was called upon.
- **Iteration**: The `reduce()` method iterates over each element of the array, applying the callback function to the accumulator and the current element.

- **Return Value**: After all elements have been processed, the `reduce()` method returns the accumulated result.

10.How does the reduceRight() method differ from the reduce() method?

The `reduceRight()` method in JavaScript is similar to the `reduce()` method, but it processes the array elements from right to left instead of left to right. Here's a more detailed explanation:

`reduce()` Method

- **Direction**: Left-to-right.
- **Syntax**: `array.reduce(callback(accumulator, currentValue, currentIndex, array), initialValue)`
- **Callback Function**: The callback function takes four arguments:
    a. `accumulator` - the accumulated value previously returned in the last invocation of the callback, or `initialValue`, if supplied.
    b. `currentValue` - the current element being processed in the array.
    c. `currentIndex` - the index of the current element being processed in the array.
    d. `array` - the array `reduce` was called upon.
- **Initial Value**: An optional argument that specifies the initial value for the accumulator. If not provided, the first element of the array is used as the initial accumulator value, and the iteration starts from the second element.

`reduceRight()` Method

- **Direction**: Right-to-left.
- **Syntax**: `array.reduceRight(callback(accumulator, currentValue, currentIndex, array), initialValue)`
- **Callback Function**: The callback function takes the same four arguments as `reduce()`.
- **Initial Value**: Works the same as in `reduce()`, but the iteration starts from the last element of the array instead of the first.