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:

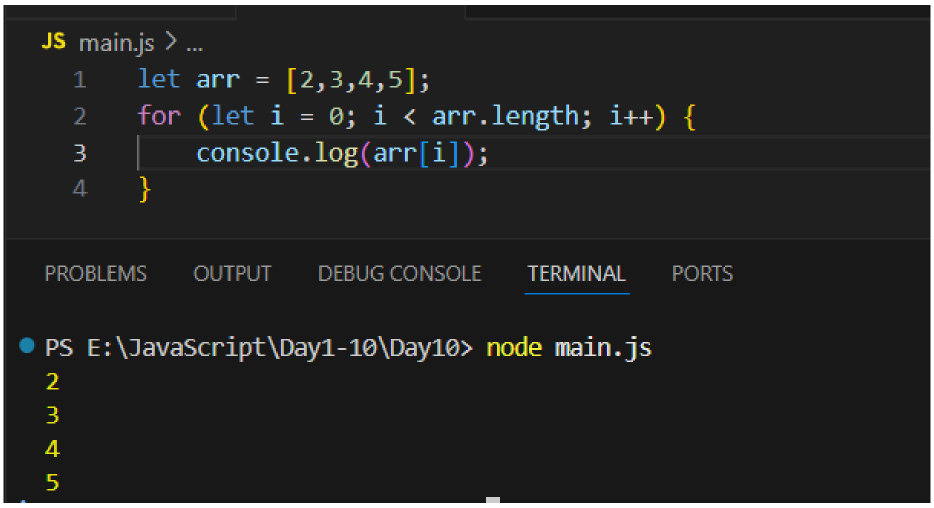
- Access each element one by one
- Perform an operation on each element
- Search, modify, or calculate something

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

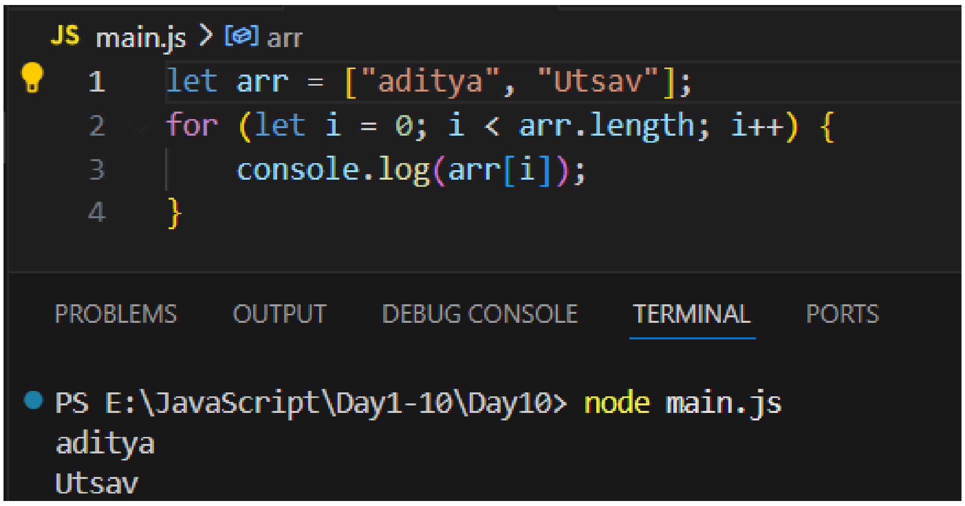
Example: to print the elements of the array using the for...loop



```javascript
let arr = [2,3,4,5];
for (let i = 0; i < arr.length; i++) {
    console.log(arr[i]);
}
```

```
PS E:\JavaScript\Day1-10\Day10> node main.js
2
3
4
5
```

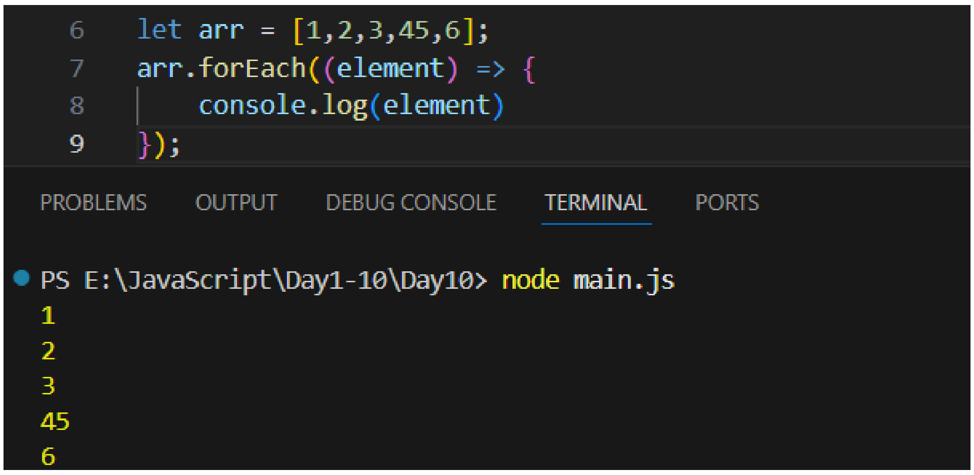
Example: to print the element where the elements of the array be the strings.



```javascript
let arr = ["aditya", "Utsav"];
for (let i = 0; i < arr.length; i++) {
    console.log(arr[i]);
}
```

```
PS E:\JavaScript\Day1-10\Day10> node main.js
aditya
Utsav
```

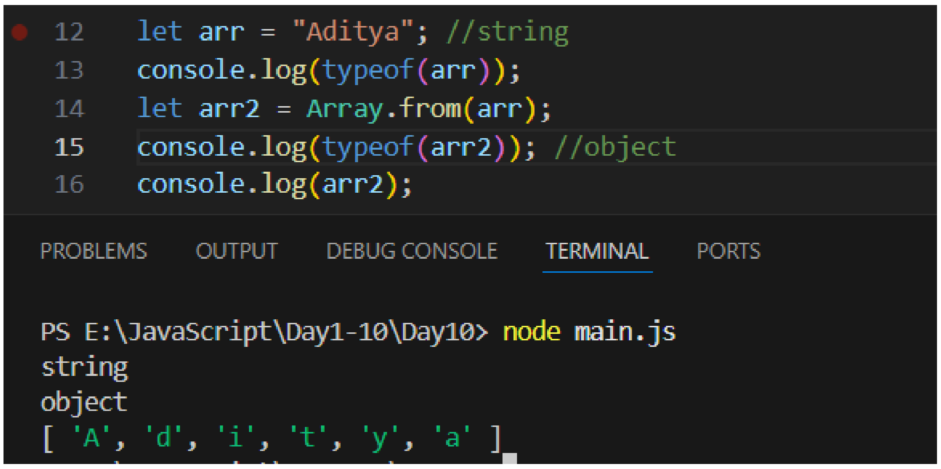
Example: we can also use forEach loop to print the elements of the array



```javascript
let arr = [1,2,3,45,6];
arr.forEach((element) => {
    console.log(element)
});
```

```
PS E:\JavaScript\Day1-10\Day10> node main.js
1
2
3
45
6
```

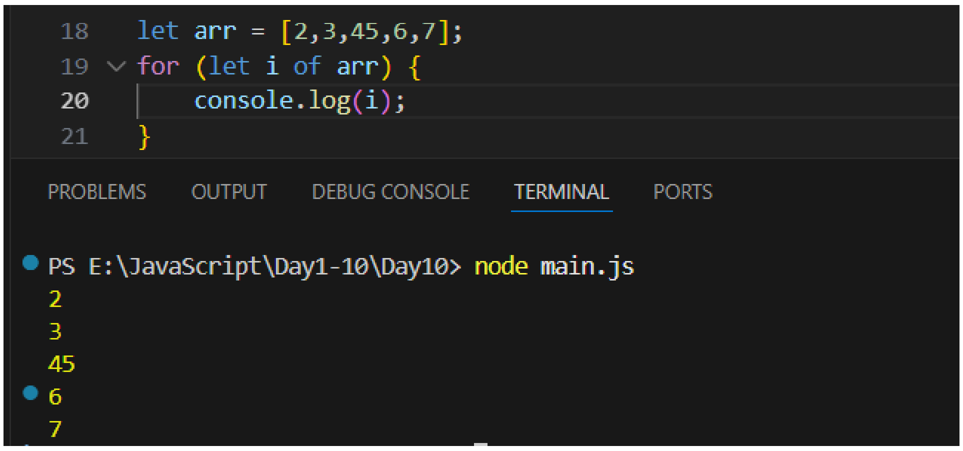
Example: Arrya.from is used to create an array from any other object.



```javascript
let arr = "Aditya"; //string
console.log(typeof(arr));
let arr2 = Array.from(arr);
console.log(typeof(arr2)); //object
console.log(arr2);
```

```
PS E:\JavaScript\Day1-10\Day10> node main.js
string
object
[ 'A', 'd', 'i', 't', 'y', 'a' ]
```

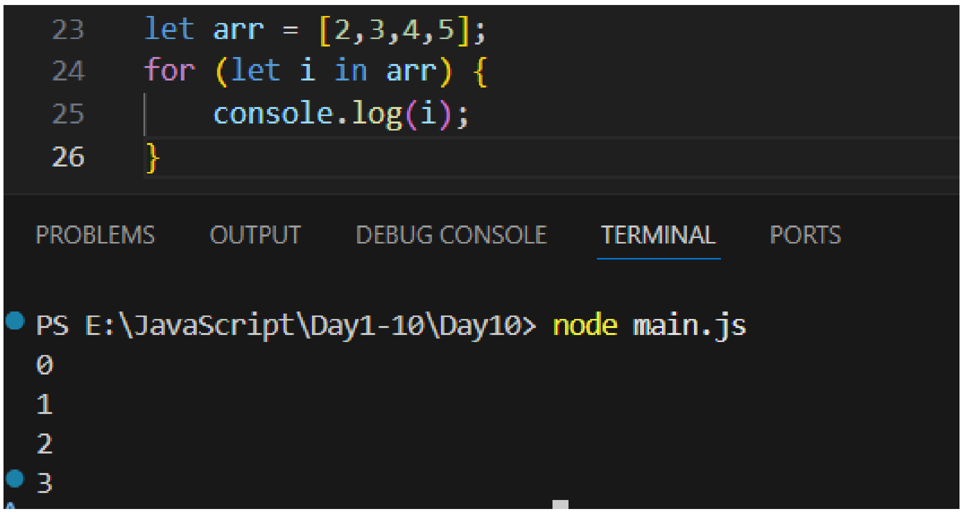
Example: for..of loop and array



```javascript
let arr = [2,3,45,6,7];
for (let i of arr) {
    console.log(i);
}
```

```
PS E:\JavaScript\Day1-10\Day10> node main.js
2
3
45
6
7
```

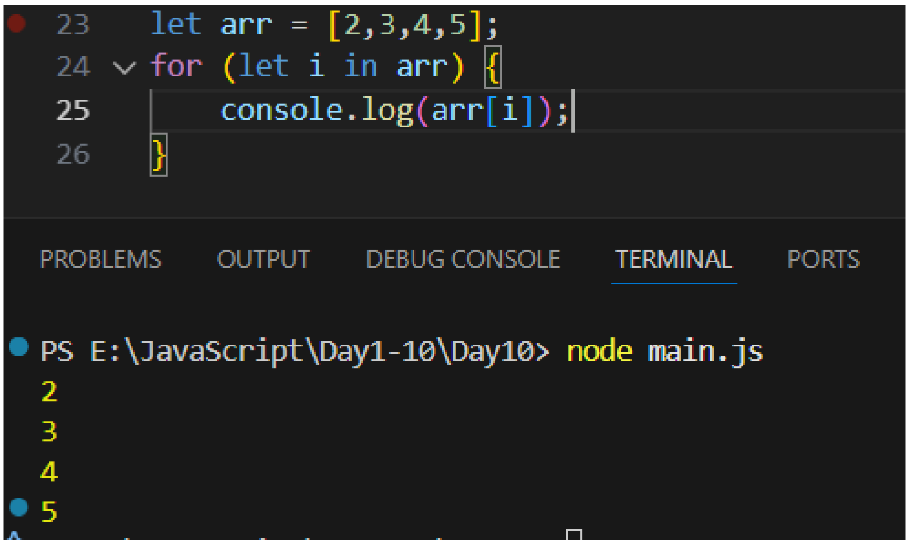
Example: for … in loop and array



```javascript
let arr = [2,3,4,5];
for (let i in arr) {
    console.log(i);
}
```

```
PS E:\JavaScript\Day1-10\Day10> node main.js
0
1
2
3
```

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



```
let arr = [2,3,4,5];
for (let i in arr) {
    console.log(arr[i]);
}
```

```
PS E:\JavaScript\Day1-10\Day10> node main.js
2
3
4
5
```

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.

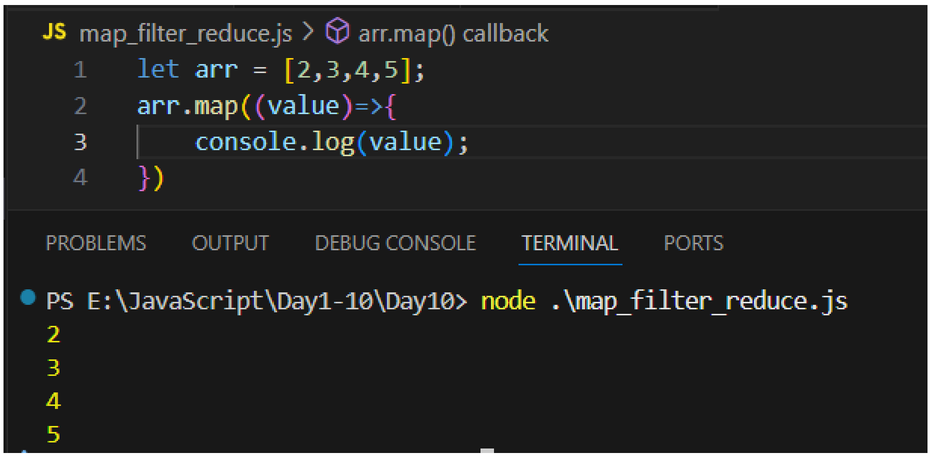
Syntax:

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

// return new value for new array

})

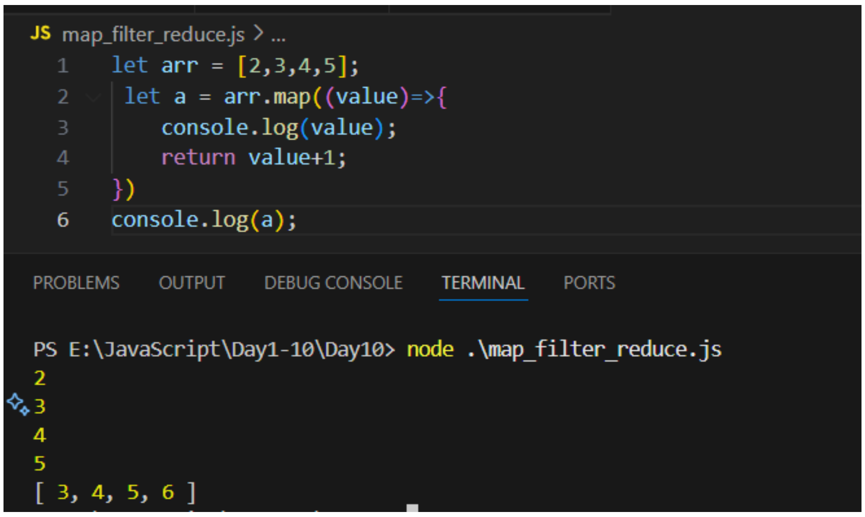
Example:



```
let arr = [2,3,4,5];
arr.map((value)=>{
    console.log(value);
})
```

```
PS E:\JavaScript\Day1-10\Day10> node .\map_filter_reduce.js
2
3
4
5
```

Example:



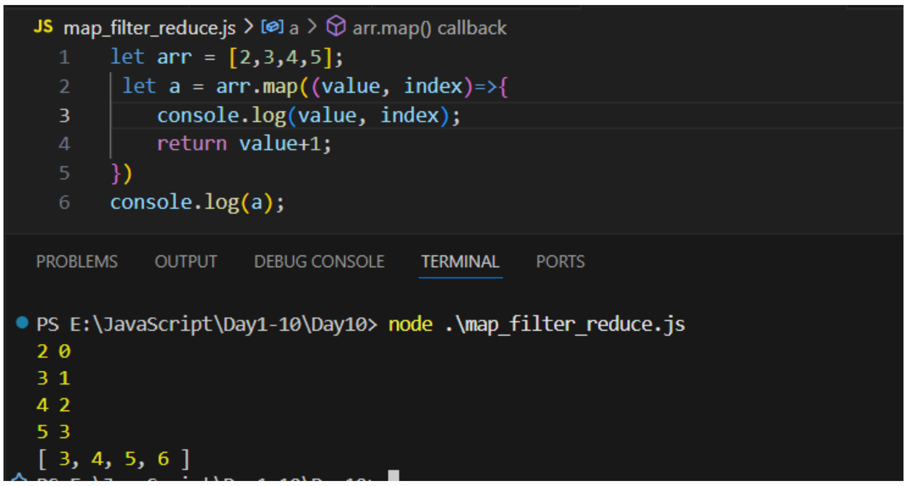
JS map_filter_reduce.js > ...

```javascript
let arr = [2,3,4,5];
let a = arr.map((value)=>{
    console.log(value);
    return value+1;
})
console.log(a);
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS E:\JavaScript\Day1-10\Day10> node .\map_filter_reduce.js
2
3
4
5
[ 3, 4, 5, 6 ]
```

Example:



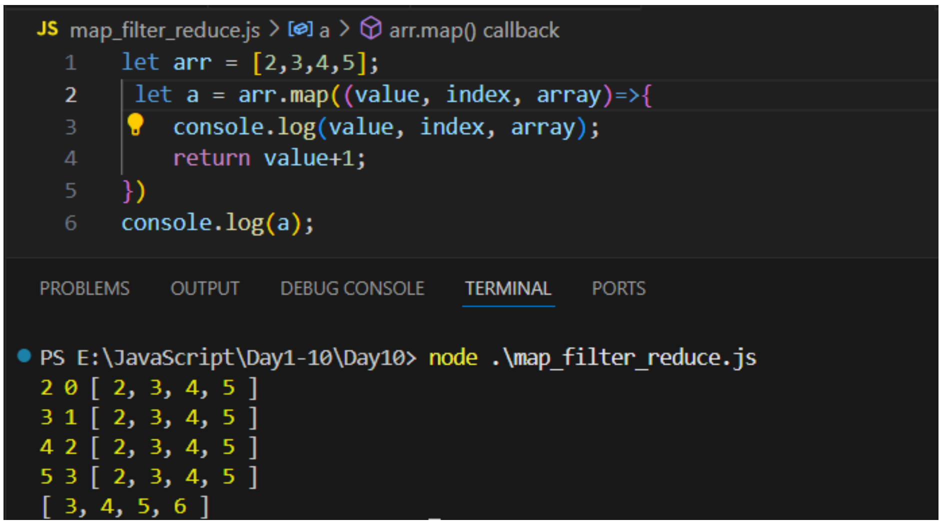
JS map_filter_reduce.js > a > arr.map() callback

```javascript
let arr = [2,3,4,5];
let a = arr.map((value, index)=>{
    console.log(value, index);
    return value+1;
})
console.log(a);
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS E:\JavaScript\Day1-10\Day10> node .\map_filter_reduce.js
2 0
3 1
4 2
5 3
[ 3, 4, 5, 6 ]
```

Example:



JS map_filter_reduce.js > a > arr.map() callback

```javascript
let arr = [2,3,4,5];
let a = arr.map((value, index, array)=>{
    console.log(value, index, array);
    return value+1;
})
console.log(a);
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS E:\JavaScript\Day1-10\Day10> node .\map_filter_reduce.js
2 0 [ 2, 3, 4, 5 ]
3 1 [ 2, 3, 4, 5 ]
4 2 [ 2, 3, 4, 5 ]
5 3 [ 2, 3, 4, 5 ]
[ 3, 4, 5, 6 ]
```

filter()

Concept:

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

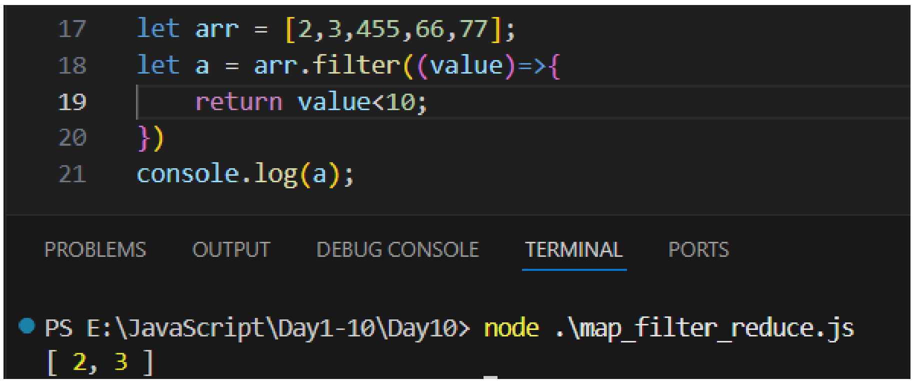
Syntax:

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

// return true to keep element, false to discard

})

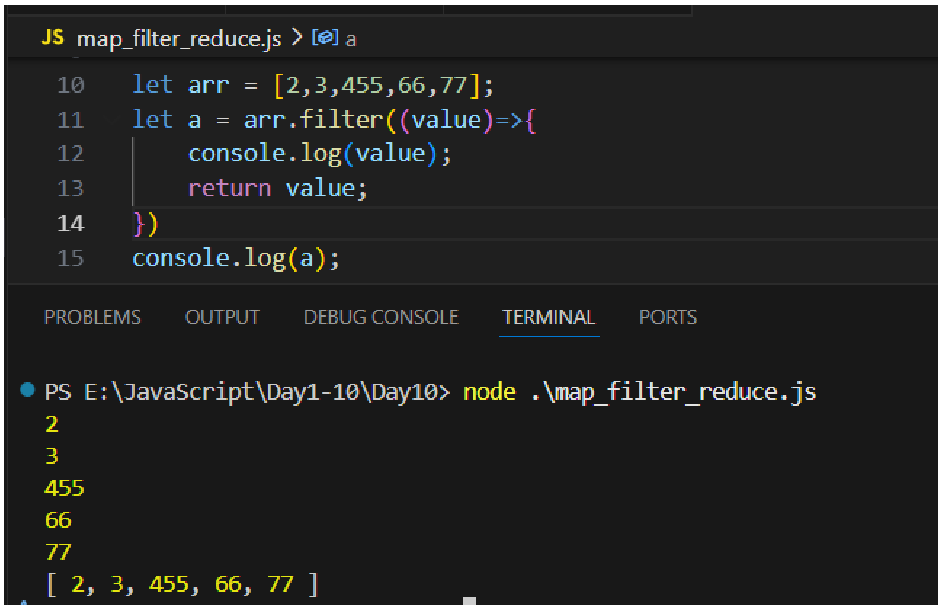
Example: to filter out those values of array whose value is less than 10.



```
let arr = [2,3,455,66,77];
let a = arr.filter((value)=>{
    return value<10;
})
console.log(a);
```

```
PS E:\JavaScript\Day1-10\Day10> node .\map_filter_reduce.js
[ 2, 3 ]
```

Example:



```
let arr = [2,3,455,66,77];
let a = arr.filter((value)=>{
    console.log(value);
    return value;
})
console.log(a);
```

```
PS E:\JavaScript\Day1-10\Day10> node .\map_filter_reduce.js
2
3
455
66
77
[ 2, 3, 455, 66, 77 ]
```

reduce()

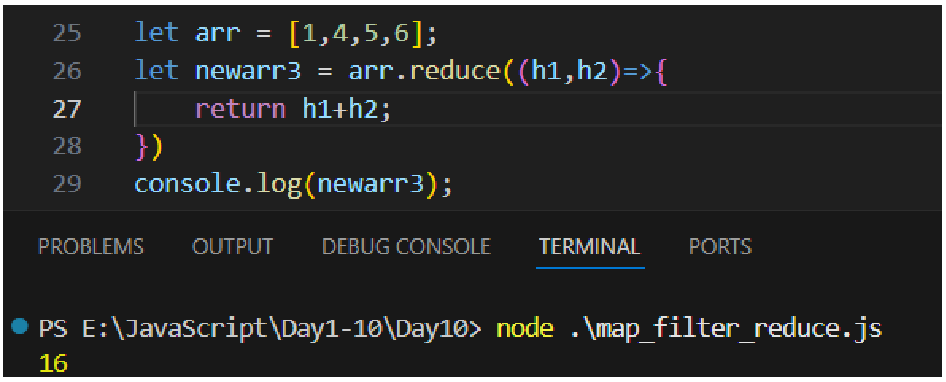
Concept:

- 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)
```

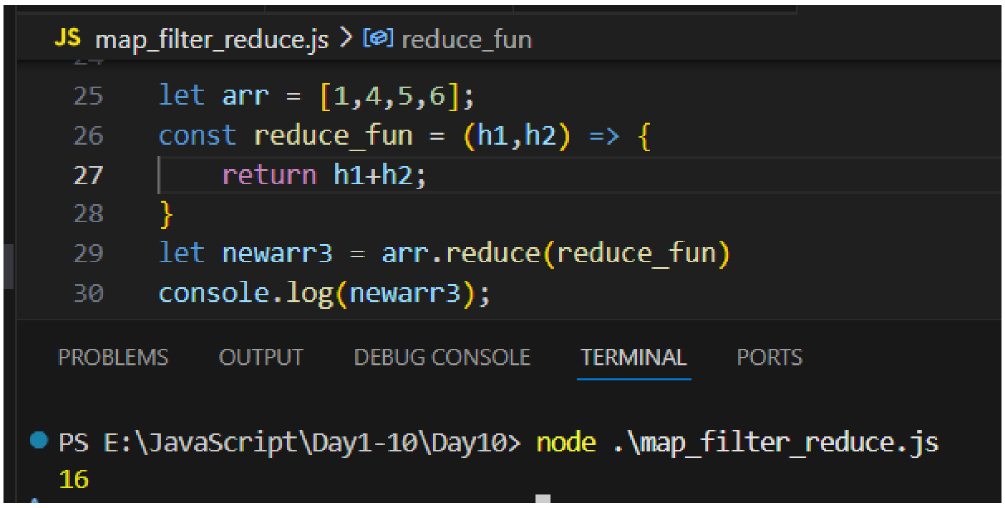
Example:



```
let arr = [1,4,5,6];
let newarr3 = arr.reduce((h1,h2)=>{
    return h1+h2;
})
console.log(newarr3);
```

```
PS E:\JavaScript\Day1-10\Day10> node .\map_filter_reduce.js
16
```

Example:



```
let arr = [1,4,5,6];
const reduce_fun = (h1,h2) => {
    return h1+h2;
}
let newarr3 = arr.reduce(reduce_fun)
console.log(newarr3);
```

```
PS E:\JavaScript\Day1-10\Day10> node .\map_filter_reduce.js
16
```

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--