

Week — 6 JS Assignment Questions And Solutions

Question 1: Reverse an Array

Problem: Write a function that takes an array and returns a new array with the elements in reverse order.

Let input = [1,2,3,4,5]

Let output = [5,4,3,2,1]



```
JS dummy.js X
JS dummy.js > ...
1 //Question 1:Reverse An Array//
2
3 function inverttheArray(arr){
4     return arr.slice().reverse()
5 }
6
7 let input = [1,2,3,4,5]
8 let output = inverttheArray(input)
9 console.log(output)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
● PS C:\Users\IBRAHIM\Desktop\PRA\HTML> node dummy.js
[ 5, 4, 3, 2, 1 ]
○ PS C:\Users\IBRAHIM\Desktop\PRA\HTML> █
```

Solution - Brief Explanation:

- ❑ **Function Definition:** The function `inverttheArray` is defined to take one parameter, `arr`, which is the input array.
- ❑ **Loop:** It uses a loop that starts from the last element of the array and goes to the first element.
- ❑ **Pushing Elements:** Each element is pushed into a new array called `reversedArr`.
- ❑ **Return:** Finally, the function returns the new array that contains the elements in reverse order.

Output: [5, 4, 3, 2, 1]



Edit with WPS Office

Question 2: Flatten an Array

Problem: Write a function that takes a nested array and flattens it to a single-level array.

Input: [1, [2, 3], [4, [5]]]

Output: [1, 2, 3, 4, 5]



```
JS dummy.js x
JS dummy.js > ...
1 //Question 2: Flatten an Array//
2 
3 const input = [1,[2,3],[4,[5]]]
4 
5 const output = input.flat(Infinity);
6 
7 console.log(output)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
● PS C:\Users\IBRAHIM\Desktop\PRA\HTML> node dummy.js
[ 1, 2, 3, 4, 5 ]
○ PS C:\Users\IBRAHIM\Desktop\PRA\HTML> 
```

Solution - Brief Explanation:

- ❑ Using flat(): The flat() method simplifies the process by flattening the array to the specified depth (in this case, Infinity means all levels).

Output: [1, 2, 3, 4, 5]



Edit with WPS Office

Question 3: Check for Duplicates

Problem: Write a function that checks if an array contains duplicates.

Input: [1, 2, 3, 4, 5, 1]

Output: true

Input: [1, 2, 3, 4, 5]

Output: false



```
JS dummy.js X
JS dummy.js > ...
1 //Question 3: Check for Duplicates//
2
3 function hasDuplicate(arr){
4     return new Set(arr).size !== arr.length;
5 }
6
7
8 const input1 = [1,2,3,4,5,1];
9 const input2 = [1,2,3,4,5,];
10
11
12 console.log(hasDuplicate(input1));
13 console.log(hasDuplicate(input2));
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
● PS C:\Users\IBRAHIM\Desktop\PRA\HTML> node dummy.js
true
false
○ PS C:\Users\IBRAHIM\Desktop\PRA\HTML> 
```

Solution - Brief Explanation:

- ❑ `new Set(arr)` creates a Set from the array `arr`. A Set only stores unique values, so if there are duplicates in `arr`, the Set will have fewer elements than `arr`.
- ❑ `Set.size` returns the number of unique elements in the Set.
- ❑ `arr.length` returns the number of elements in the original array.
- ❑ By comparing `new Set(arr).size` with `arr.length`, you can determine if there are duplicates:



Edit with WPS Office

- ❑ If the sizes are different, there are duplicates, and the function returns true.
- ❑ If the sizes are the same, there are no duplicates, and the function returns false.

Output: true

Output: true

Question 4: Merge Two Objects

Problem: Write a function that merges two objects into one.

Input: { a: 1, b: 2 }, { b: 2, c: 4 }

Output: { a: 1, b: 2, c: 4 }



```
JS dummy.js X
JS dummy.js > mergeObjects > obj2
1 //Question 4: Merge two objects//
2
3 function mergeObjects(obj1,obj2){
4     return {...obj1,...obj2}
5 }
6
7 const object1 = {a:1,b:2};
8 const object2 = {b:2,c:4};
9
10 const mergedObject = mergeObjects(object1,object2);
11 console.log(mergedObject);
12
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
● PS C:\Users\IBRAHIM\Desktop\PRA\HTML> node dummy.js
{ a: 1, b: 2, c: 4 }
○ PS C:\Users\IBRAHIM\Desktop\PRA\HTML>
```

Solution - Brief Explanation:

- The ... (spread) operator is used to merge the properties of obj1 and obj2.
- If there are overlapping keys (like b in this example), the value from obj2 will override the value from obj1.
- **Output: { a: 1, b: 2, c: 4 }**



Edit with WPS Office

Question 5: Find the Maximum Number in an Array

Problem: Write a function that finds the maximum number in an array.

Input: [1, 3, 2, 8, 5]

Output : 8

```
JS dummy.js X
JS dummy.js > ...
1  //Question 5: Find the Maximum Number in an Array//
2
3  function findMaxNumber(arr){
4      return Math.max(...arr);
5  }
6
7
8  const ArrayNumbers = [1,3,2,8,5];
9  console.log(findMaxNumber(ArrayNumbers));
10
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

● PS C:\Users\IBRAHIM\Desktop\PRA\HTML> node dummy.js
8

○ PS C:\Users\IBRAHIM\Desktop\PRA\HTML>



Edit with WPS Office

Solution - Brief Explanation:

- ❑ `Math.max(...arr)` uses the spread operator `...` to spread the elements of `arr` and pass them individually to `Math.max()`, which returns the largest value.
- ❑ `const ArrayNumbers = [1,3,2,8,5];` defines the array of numbers.
- ❑ `console.log(findMaxNumber(ArrayNumbers));` calls the `findMaxNumber` function and logs the maximum number to the console.

Output: 8

Question 6: Group Array of Objects by Property

Problem: Write a function that groups an array of objects by a specific property.

Input: [{ id: 1, category: 'fruit' }, { id: 2, category: 'vegetable' }, { id: 3, category: 'fruit' }]

Output: {

fruit: [{ id: 1, category: 'fruit' }, { id: 3, category: 'fruit' }],

vegetable: [{ id: 2, category: 'vegetable' }]



}

```
JS dummyjs X
JS dummyjs > ...
1 //Question :6 Group Array of Objects by Property//
2 function groupByProperty(arr, prop) {
3     return arr.reduce((acc, obj) => {
4         // Initialize the array for the property if it doesn't exist and push the object into it
5         (acc[obj[prop]] = acc[obj[prop]] || []).push(obj);
6         return acc;
7     }, {}); // Start with an empty object
8 }
9
10 // Example usage:
11 const input = [
12     { id: 1, category: 'fruit' },
13     { id: 2, category: 'vegetable' },
14     { id: 3, category: 'fruit' }
15 ];
16
17 console.log("Grouped Objects:", groupByProperty(input, 'category'));

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
● PS C:\Users\IBRAHIM\Desktop\PRA\HTML> node dummy.js
Grouped Objects: {
  fruit: [ { id: 1, category: 'fruit' }, { id: 3, category: 'fruit' } ],
  vegetable: [ { id: 2, category: 'vegetable' } ]
}
○ PS C:\Users\IBRAHIM\Desktop\PRA\HTML>
```

Solution - Brief Explanation:

Purpose: The function groups an array of objects based on a specified property.

Parameters: arr: The array of objects to be grouped.

prop: The property name used for grouping.

Using reduce:

reduce is a method used to iterate over each object in arr and accumulate results in an object.



Edit with WPS Office

Logic:

`(acc[obj[prop]] = acc[obj[prop]] || []):`

This line checks if `acc[obj[prop]]` (the array for the specific property) already exists.

If it doesn't, it initializes it as an empty array (`[]`).

`.push(obj)`: Adds the current object `obj` to the appropriate array.

5. Result:

The objects are grouped into arrays based on the value of `prop`, and the final grouped object is returned.

Output:

{

fruit: [{ id: 1, category: 'fruit' }, { id: 3, category: 'fruit' }],

vegetable: [{ id: 2, category: 'vegetable' }]

}



Edit with WPS Office

Question 7: Find the Intersection of Two Arrays

Problem: Write a function that returns the intersection of two arrays.

Input: [1, 2, 3], [2, 3, 4]

Output: [2, 3]

```
JS dummy.js X
JS dummy.js > ...
1  //Question :7 Find the Intersection of the two Arrays//
2
3  function intersectArrays(arr1,arr2){
4      return arr1.filter(value => arr2.includes(value));
5  }
6
7
8  const array1 = [1,2,3];
9  const array2 = [2,3,4];
10
11 const output = intersectArrays(array1,array2);
12 console.log(output);
13
14
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
● PS C:\Users\IBRAHIM\Desktop\PRA\HTML> node dummy.js
[ 2, 3 ]
○ PS C:\Users\IBRAHIM\Desktop\PRA\HTML> |
```



Edit with WPS Office

Solution - Brief Explanation:

- arr1 is [1, 2, 3].
- filter starts by looking at the first element of arr1, which is 1.

It checks arr2.includes(1), which is false because 1 is not in arr2 ([2, 3, 4]). So, 1 is not included in the new array.

- Then it checks the second element of arr1, which is 2.

It checks arr2.includes(2), which is true because 2 is in arr2. So, 2 is included in the new array.

- Finally, it checks the third element of arr1, which is 3.

It checks arr2.includes(3), which is true because 3 is also in arr2. So, 3 is included in the new array.

- The filter method then returns the new array [2, 3].

Output :[2,3]

Question 8: Calculate the Sum of Array Elements



Edit with WPS Office

Problem: Write a function that calculates the sum of all numbers in an array.

Input: [1, 2, 3, 4, 5]

Output: 15

```
JS dummy.js X
JS dummy.js > ...
1 //Question :8 Calculate The sum of Array Elements//
2
3 function sumArray(arr) {
4     return arr.reduce((accumulator, currentValue) => accumulator + currentValue, 0);
5 }
6
7 // Example usage:
8 const inputArray = [1, 2, 3, 4, 5];
9 const result = sumArray(inputArray);
10 console.log(result); // Output: 15
11
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\IBRAHIM\Desktop\PRA\HTML> node dummy.js
15
PS C:\Users\IBRAHIM\Desktop\PRA\HTML>
```

Solution - Brief Explanation:

- ❑ **Function Definition:** The function `sumArray` takes one parameter, `arr`, which is the array of numbers.
- ❑ **Using reduce:** `arr.reduce(...)` goes through each element of the array and applies the provided function to accumulate a result.



Edit with WPS Office

- The function `(accumulator, currentValue) => accumulator + currentValue` takes two parameters:
- `accumulator`: This keeps track of the ongoing sum.
- `currentValue`: This represents the current element being processed from the array. The 0 at the end is the initial value for the accumulator, starting the sum from zero.

Step-by-Step for the Example Input

1. Start with `accumulator = 0`.
2. Add the first element (1): $0 + 1 = 1$.
3. Add the second element (2): $1 + 2 = 3$.
4. Add the third element (3): $3 + 3 = 6$.
5. Add the fourth element (4): $6 + 4 = 10$.
6. Add the fifth element (5): $10 + 5 = 15$.
7. The final result is 15.

Question 9: Remove Falsy Values from an Array

Problem: Write a function that removes all falsy values from an array.

Input: `[0, 1, false, 2, "", 3]`



Edit with WPS Office

Output: [1, 2, 3]

```
JS dummy.js X
JS dummy.js > ...
1 // Question 9: Remove Falsy Values from an Array//
2
3 function removeFalsyValues(arr) {
4   return arr.filter(Boolean);
5 }
6
7 // Example usage:
8 const input = [0, 1, false, 2, '', 3];
9 const output = removeFalsyValues(input);
10 console.log(output); // Output: [1, 2, 3]
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

● PS C:\Users\IBRAHIM\Desktop\PRA\HTML> node dummy.js
[1, 2, 3]

○ PS C:\Users\IBRAHIM\Desktop\PRA\HTML> █

Solution - Brief Explanation:

- ❑ Step 1: We have an input array with values like 0, 1, false, 2, "", and 3.
- ❑ Step 2: We use the filter method to check each element in the input array. By passing Boolean as the function, it removes all falsy values.
- ❑ Step 3: We print the output array, which now only includes the truthy values: [1, 2, 3].



Edit with WPS Office

Boolean

Boolean is a built-in function in JavaScript that checks whether each value in the array is truthy or falsy.

Meaning of Boolean in This Code

- ❑ When we use Boolean inside the filter method:
- ❑ It checks each element in the array to determine if it is truthy or falsy.
- ❑ If Boolean(element) returns true, the element stays in the array.
- ❑ If Boolean(element) returns false, the element is removed from the array.

Example Explanation

- ❑ Boolean(0) → false (0 is falsy, so it's removed)
- ❑ Boolean(1) → true (1 is truthy, so it's kept)



- `Boolean(false)` → `false` (`false` is falsy, so it's removed)
- `Boolean(2)` → `true` (`2` is truthy, so it's kept)
- `Boolean("")` → `false` (`""` is falsy, so it's removed)
- `Boolean(3)` → `true` (`3` is truthy, so it's kept)

Falsy Value

Falsy Value is a value in JavaScript that evaluates to `false` when converted to a `Boolean`.

List of Falsy Values

Here are the common falsy values in JavaScript:

1. `false` — The `Boolean` value `false`.
2. `0` — The number zero.



3. -0 — Negative zero (though its still zero, its considered falsy).
4. " or "" — An empty string (both single or double quotes).
5. null — The absence of any value or object.
6. undefined — A variable that has been declared but not assigned a value.
7. NaN — The special "Not-a-Number" value.

Question 10: Calculate Average of an Array

Problem: Write a function that calculates the average of all numbers in an array.

Input: [1, 2, 3, 4, 5]

```
JS dummyjs X
JS dummyjs > ...
1 // Question 10: Caculate Average of an Array //
2
3 function calculateAverage(arr) {
4     return arr.length ? arr.reduce((sum, num) => sum + num, 0) / arr.length : 0;
5 }
6
7 // Example usage
8 const input = [1,2,3,4,5]
9 console.log(calculateAverage([1,2,3,4,5])); // Output: 3
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\IBRAHIM\Desktop\PRA\HTML> node dummy.js
3



Edit with WPS Office

Solution - Brief Explanation:

- ❑ **Function Definition:** `calculateAverage(arr)` takes an array of numbers as input.
- ❑ **2. Check for Empty Array:** Uses a conditional (`arr.length ? ... : 0`) to return 0 if the array is empty.
- ❑ **3. Sum Calculation:** Utilizes `reduce` to accumulate the sum of the numbers, starting from an initial value of 0.
- ❑ **4. Average Calculation:** Divides the total sum by the length of the array to compute the average.
- ❑ **5. Return Value:** Returns the calculated average or 0 if the array is empty.
- ❑ **6. Example Usage:**
- ❑ **Input:** `[1, 2, 3, 4, 5]`

Output: 3

