Array - JavaScript Coding Interview Questions with Answer

1. **Find the Largest Number in an Array: Write a function to find the largest number in an array.**
   1. **Using Traditional Approach**

```
function maximum(array) {

    if(array.length === 0)
        return 'Array is empty';

        let max = array[0];

    for(let i=0; i < array.length; i++) {
        if(array[i] > max) {
            max = array[i];
        }
    }
    return max;
}

let arr = [10, 21, 8, 50, 7, 25];
let arr1 = [];
console.log(maximum(arr));
```

   2. **Using 'sort()' method:**

```
function maximum(array) {

    if(array.length === 0)
        return 'Array is empty';

    array.sort((a, b) => a-b );
    return array[array.length-1];
}

let arr = [10, 21, 8, 50, 7, 25];
console.log(maximum(arr));
```

   3. **Using reduce method:**

```
function maximum(array) {

    if(array.length === 0)
        return 'Array is empty';

    return array.reduce((max, num) => num > max ? num : max);
}

let arr = [10, 21, 8, 50, 7, 25];
console.log(maximum(arr));
```

   4. **Using Math.max() method**

```
function maximum(array) {

    if(array.length === 0)
```

```
        return 'Array is empty';

    return Math.max(...array);
}

let arr = [10, 21, 8, 50, 7, 25];
console.log(maximum(arr));
```

2. Find the Second Largest Number in an Array: Write a function to find the second largest number in an array.

1. Using Traditional Approach

```
function findSecondLargest(arr) {
    if (arr.length < 2) {
        return null;
    }

    let firstMax = arr[0];
    let secondMax = Number.MIN_SAFE_INTEGER;

    for (let i = 1; i < arr.length; i++) {
        if (arr[i] > firstMax) {
            secondMax = firstMax;
            firstMax = arr[i];
        } else if (arr[i] > secondMax && arr[i] !== firstMax) {
            secondMax = arr[i];
        }
    }
    return secondMax !== Number.MIN_SAFE_INTEGER ? secondMax : null;
}

let arr = [10, 5, 20, 45, 8, 24, 44];
console.log(findSecondLargest(arr));
```

2. Using 'sort()' method:

```
function findSecondLargest(arr) {
    if (arr.length < 2) {
        return null;
    }

    arr.sort((a, b) => b - a ); // descending order

    return arr[1];
}

let arr = [10, 5, 20, 45, 8, 24, 44];
console.log(findSecondLargest(arr));
```

3. using Set and Math.max:

```
function findSecondLargest(arr) {
    if (arr.length < 2) {
        return "Invalid Input";
    }
```

```
    const uniqueArray = [... new Set(arr)];
    const max = Math.max(...uniqueArray);
    uniqueArray.splice(uniqueArray.indexOf(max), 1);
    return Math.max(...uniqueArray);
}

let arr = [10, 5, 20, 45, 8, 24, 44];
console.log(findSecondLargest(arr));
```

4. Using Sorting with Set:

Convert the array to a Set to remove duplicates, sort the resulting array, and return the second last element.

```
function findSecondLargest(arr) {
    const uniqueArray = [...new Set(arr)];
    uniqueArray.sort((a, b) => b - a);
    return uniqueArray[1];
}

let arr = [10, 24, 54, 45, 12, 8];
console.log(findSecondLargest(arr));
```

5. Find the Nth Largest Number in an Array: Write a function to find the Nth largest number in an array.

You can find the Nth largest number in an array by sorting the array in descending order and then accessing the Nth element from the sorted array.

```
function findNthLargest(arr, n) {
    // sort the array in descending array
    arr.sort((a, b) => b-a);

    return arr[n-1];
}

let arr = [10, 24, 15, 31, 20, 55];
const n = 3;
console.log(findNthLargest(arr, n));
```

6. Find the Smallest Number in an Array: Write a function to find the smallest number in an array.

1. Using Traditional Approach

```
function findSmallest(arr) {
   if(arr.length < 0) {
    return "Array is Empty";
   }
   let min = arr[0];
   for(let i=0; i < arr.length; i++) {
    if(arr[i] < min) {
        min = arr[i];
    }
   }
   return min;
}

let arr = [10, 5, 3, 20, 45, 8, 24, 44];
console.log(findSmallest(arr));
```

2. Using sort method

```
function findSmallest(arr) {
   arr.sort((a, b) => a-b);
   return arr[0];
}

let arr = [10, 5, 3, 20, 45, 8, 24, 44];
console.log(findSmallest(arr));
```

3. Using Set and Math.min method

```
function findSmallest(arr) {
    const uniqueArray = [... new Set(arr)];
    const min = Math.min(...uniqueArray);
   return min;
}

let arr = [10, 5, 20, 45, 8, 24, 44];
console.log(findSmallest(arr));
```

4. Find the Second Smallest Number in an Array: Write a function to find the second smallest number in an array.

```
function findSecondSmallest(arr) {
    const uniqueArray = [...new Set(arr)];
    uniqueArray.sort((a, b) => a-b);
    return uniqueArray[1];
}

let arr = [10, 5, 3, 20, 45, 8, 24, 44];
console.log(findSecondSmallest(arr));
```

5. Find the Nth Smallest Number in an Array: Write a function to find the Nth smallest number in an array.

```
function findSecondSmallest(arr, n) {
    const uniqueArray = [...new Set(arr)];
    uniqueArray.sort((a, b) => a-b);
    return uniqueArray[n-1];
}

let arr = [10, 5, 3, 20, 45, 8, 24, 44];
const n = 3;
console.log(findSecondSmallest(arr, n));
```

6. Find Duplicates in an Array: Write a function to find all the duplicate elements in an array.

```
function findDuplicates(arr) {
    const duplicates = {};
    const result = [];

    for (const element of arr) {
        if (duplicates[element]) {
            duplicates[element]++;
        } else {
            duplicates[element] = 1;
```

```
        }
    }

    for (const key in duplicates) {
        if (duplicates[key] > 1) {
            result.push(key);
        }
    }

    return result;
}

const numbers = [1, 2, 3, 2, 4, 5, 4, 6, 7, 8, 7, 9];
console.log(findDuplicates(numbers)); // Output: [ '2', '4', '7' ]
```

7. **Remove Duplicates from an Array: Write a function to remove all duplicate elements from an array.**
8. **Rotate Array Elements: Write a function to rotate the elements of an array to the right by a given number of steps.**
9. **Find Missing Number in an Array: Write a function to find the missing number in an array of integers from 1 to n.**
10. Sum of Two Numbers: Given an array of integers and a target sum, write a function that returns true if there are two numbers in the array that add up to the target sum, false otherwise.
11. Find Intersection of Two Arrays: Write a function that returns an array containing all the elements that are common to both input arrays.
12. Merge Sorted Arrays: Write a function that merges two sorted arrays into a single sorted array.
13. Find Maximum Consecutive Ones: Given a binary array, find the maximum number of consecutive 1s in this array.
14. Find Maximum Product of Two Integers in an Array: Write a function to find the maximum product of two integers in an array.

1. **How can you double elements of an array? Do not use extra variable?**

```
/*How can you double elements of an array? Do not use extra variable?*/

let arr = [11, 12, 13, 14, 15];

// Traditional Approach
for(let i = 0; i < arr.length; i++) {
    arr[i] = arr[i] * 2;
}
console.log(arr);

// Using map method
arr = arr.map((item) => item*2);*/

// Using forEach()
arr.forEach((item, index, arr) => arr[index] *= 2);
console.log(arr);
```

```
// reduce method is not recommended way to double element of array.
/*arr.reduce((accumulator, currentValue, index, array)=> {
    array[index-1] =array[index-1] *2
});*/
console.log(arr);
```

2. How to check if given input is string?

1. Using the typeof operator:

```
function isString(input) {
    return typeof input === 'string';
}

console.log(isString("hello"));  // Output: true
console.log(isString(123));      // Output: false
```

2. Using the instanceof operator:

```
function isString(input) {
    return input instanceof String || typeof input === 'string';
}

console.log(isString("hello"));  // Output: true
console.log(isString(123));      // Output: false
```

The typeof operator returns a string indicating the type of the unevaluated operand. When used with a string, it returns 'string'. However, it's important to note that when checking string primitives, typeof will return 'string', but when checking String objects created with new String(), it will return 'object'. Hence, the instanceof operator is used to cover both cases. It checks whether the input is an instance of the String object or if it's a string primitive.

Either of these methods can be used to check if a given input is a string in JavaScript.

3. How to check if given input is array?

1. Using the Array.isArray() method:

```
function isArray(input) {
    return Array.isArray(input);
}

console.log(isArray([1, 2, 3]));  // Output: true
console.log(isArray("hello"));    // Output: false
console.log(isArray({}));         // Output: false
```

2. Using instanceof operator

```
function isArray(input) {
    return input instanceof Array;
}

console.log(isArray([1, 2, 3]));  // Output: true
console.log(isArray("hello"));    // Output: false
console.log(isArray({}));         // Output: false
```

4. What will be the result when executing the given code?

```
const arr = ['A', 'N', 'U'];
arr[10] = 10;
console.log(arr.length);
```

Output:
11
Explanation:
The array arr is initialized with elements and an assignment at index 10. Despite gaps in indices, arr.length reflects the highest index plus one. Thus, the output is '11'.

5. What is the output of the following code?

```
let array = [1, 2, 3, 4, 5];
array.length = 0;
console.log(array)
```

Output:
[]
Explanation:
The code sets the length of the array to 0 using array.length = 0, effectively clearing all elements from the array. As a result, when you log the array to the console (console.log(array)), you'll get an empty array ([]).

6. How can you extract and print the unique values from an array?

1. Using a Set:

```
function printUniqueValues(array) {

  const uniqueValues = [...new Set(array)];
  uniqueValues.forEach(value => console.log(value));
}

const array = [1, 2, 3, 4, 2, 3, 5];
printUniqueValues(array); // Output: 1, 2, 3, 4, 5
```

Explanation:
The getUniqueValues utilises the Set data structure to automatically remove duplicate values. The spread operator (...) is then used to convert the unique values back into an array.

2. Using Array.filter and indexOf:

```
function printUniqueValues(array) {
  const uniqueValues = array.filter((value, index)=> array.indexOf(value) === index);
  uniqueValues.forEach(value => console.log(value));
}

const array = [1, 2, 3, 4, 2, 3, 5];
printUniqueValues(array); // Output: 1, 2, 3, 4, 5
```

3. Using reduce():

```
function printUniqueValues(array) {
  const uniqueValues = array.reduce((accumulator, currentValue) => {
   if(!accumulator.includes(currentValue)) {
      accumulator.push(currentValue);
   }
```

```
    return accumulator;
  }, []);
   uniqueValues.forEach(value => console.log(value));
}

const array = [1, 2, 3, 4, 2, 3, 5];
printUniqueValues(array); // Output: 1, 2, 3, 4, 5
```

7. How can you determine if a number is an integer in JavaScript?

1. **Using 'Number.isInteger( )' method:**

```
let num1 = 5;
let num2 = 5.5;
console.log(Number.isInteger(num1));  // true
console.log(Number.isInteger(num2));  // false
```

2. **Using '%' operator**

```
let num1 = 5;
let num2 = 5.5;
console.log(num1 % 1 === 0);  // true
console.log(num2 % 1 === 0);  // false
```

Explanation:
The function utilizes the modulus operator (%) to check if the remainder when dividing by 1 is zero, indicating an integer.

3. **Using 'Math.floor()' or 'Math.ceil()':**

```
let num1 = 5;
let num2 = 5.5;
console.log(Math.floor(num1) === num1 || Math.floor(num1) === num1);  // true
console.log(Math.floor(num2) === num2 || Math.floor(num2) === num2);  // false
```

Question 8:

8. What is the output of the following code?

let arr1 = [10, 12, 3.1];
let arr2 = [10, 12, 3.1];
console.log(arr1 == arr2);

Output: false
Explanation:
Arrays are reference types in JavaScript, so arr1 and arr2 point to different references, even though their contents are the same. Therefore, arr1 == arr2 will be false.

9. What is the output of the following code?

let originalArray = [1, 2, 3];
let copiedArray = originalArray.slice();

copiedArray.push(4);

console.log(originalArray);
console.log(copiedArray);

Output:
[1, 2, 3]
[1, 2, 3, 4]
Explanation:
In the above JavaScript code, originalArray and copiedArray initially reference different array objects due to the use of the slice() method, which creates a shallow copy.

Therefore, modifying copiedArray by pushing the value 4 does not affect the underlying array that originalArray references.

As a result, when you log originalArray to the console, it remains unchanged, and you will see the output [1, 2, 3]. Conversely, logging copiedArray reflects the addition of the value 4, resulting in the output [1, 2, 3, 4].

10. What is the output of the following code?

```
function checkArraysEquality(arrayA, arrayB) {
  return JSON.stringify(arrayA) === JSON.stringify(arrayB);
}

const array1 = ['a', 'b', 'c'];
const array2 = ['a', 'b', 'c'];
const array3 = ['z', 'b', 'c'];

console.log(checkArraysEquality(array1, array2));
console.log(checkArraysEquality(array3, array2));
```

Output:
true
false

Explanation:
JSON.stringify plays a key role by transforming arrays into a consistent string format, aiding in effortless comparison. This approach simplifies the equality check process and enhances code readability, making it a convenient choice for array comparison in JavaScript.

Conclusion:
In conclusion, JavaScript array interview questions often cover a range of topics related to array manipulation, iteration, and understanding various array methods. It is important to be familiar with common array operations such as adding and removing elements, iterating over arrays, checking for the existence of elements, and finding the length of an array. Additionally, knowledge of concepts like unique values and emptying arrays can also be beneficial. By having a good understanding of these array-related concepts and practicing with coding examples, you can effectively tackle JavaScript array interview questions.