Weekly Test (16-02-2025)

- **String Reversal:** Write a function to reverse a given string in JavaScript without using built-in reverse functions.

input

```
JS string_reverse.js > ...
function reverseString(str) {
    let reversed = "";
    for (let i = str.length - 1; i >= 0; i--) {
        reversed += str[i];
    }
    return reversed;
}

// Example usage:
console.log(reverseString("hello")); // Output: "olleh"
```

Output

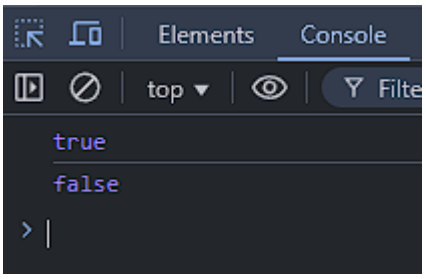
```
Elements   Console   Sou
top ▾   Filter
olleh
>
```

- **Anagram Check:** Implement an algorithm to check if two strings are anagrams of each other (contain the same characters with the same frequency)

Input

```javascript
function areAnagrams(str1, str2) {
    if (str1.length !== str2.length) {
        return false;
    }

    let charCount = {};

    // Count frequency of each character in str1
    for (let char of str1) {
        charCount[char] = (charCount[char] || 0) + 1;
    }

    // Subtract frequency using str2
    for (let char of str2) {
        if (!charCount[char]) {
            return false;
        }
        charCount[char]--;
    }

    return true;
}

// Example usage:
console.log(areAnagrams("listen", "silent")); // Output: true
console.log(areAnagrams("hello", "world"));   // Output: false
```

Output

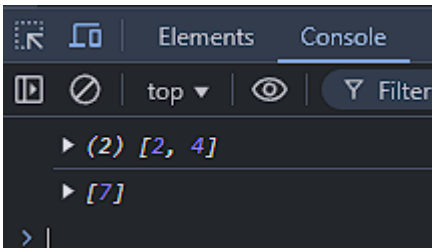


- **Array Intersection**: Given two arrays, write a function to find their intersection (common elements).

Input

```javascript
function arrayIntersection(arr1, arr2) {
    let set1 = new Set(arr1);
    let intersection = [];

    for (let num of arr2) {
        if (set1.has(num)) {
            intersection.push(num);
            set1.delete(num); // To avoid duplicate values in the result
        }
    }

    return intersection;
}

// Example usage:
console.log(arrayIntersection([1, 2, 2, 3, 4], [2, 2, 4, 6])); // Output: [2, 4]
console.log(arrayIntersection([7, 8, 9], [10, 11, 7]));       // Output: [7]
```

Output

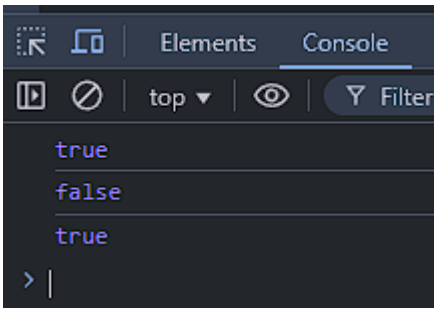


- **String Palindrome:** Create a function to check if a given string is a palindrome (reads the same forwards and backwards) while ignoring non-alphanumeric characters.

Input

```javascript
function isPalindrome(str) {
    // Convert to lowercase and remove non-alphanumeric characters
    let cleanedStr = str.toLowerCase().replace(/[^a-z0-9]/g, '');

    let left = 0, right = cleanedStr.length - 1;

    while (left < right) {
        if (cleanedStr[left] !== cleanedStr[right]) {
            return false;
        }
        left++;
        right--;
    }

    return true;
}

// Example usage:
console.log(isPalindrome("A man, a plan, a canal: Panama")); // Output: true
console.log(isPalindrome("race a car"));                     // Output: false
console.log(isPalindrome("No lemon, no melon!"));           // Output: true
```

Output

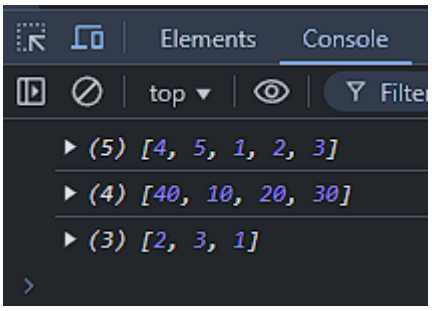


- **Array Rotation:** Implement a function to rotate an array to the right by a specified number of positions.

Input

```javascript
function rotateArray(arr, k) {
    let n = arr.length;
    if (n === 0) return arr;

    k = k % n; // Handle cases where k > n

    return [...arr.slice(-k), ...arr.slice(0, n - k)];
}

// Example usage:
console.log(rotateArray([1, 2, 3, 4, 5], 2)); // Output: [4, 5, 1, 2, 3]
console.log(rotateArray([10, 20, 30, 40], 1)); // Output: [40, 10, 20, 30]
console.log(rotateArray([1, 2, 3], 5)); // Output: [2, 3, 1]
```

Output

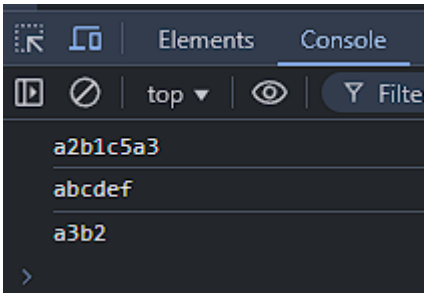


- **String Compression:** Write a function to perform basic string compression using the counts of repeated characters. For example, "aabcccccaaa" would become "a2b1c5a3."

Input

```javascript
function compressString(str) {
    let compressed = "";
    let count = 1;

    for (let i = 0; i < str.length; i++) {
        if (str[i] === str[i + 1]) {
            count++;
        } else {
            compressed += str[i] + count;
            count = 1;
        }
    }

    return compressed.length < str.length ? compressed : str;
}

// Example usage:
console.log(compressString("aabcccccaaa")); // Output: "a2b1c5a3"
console.log(compressString("abcdef"));      // Output: "abcdef" (compression wouldn't reduce size)
console.log(compressString("aaabb"));       // Output: "a3b2"
```

Output

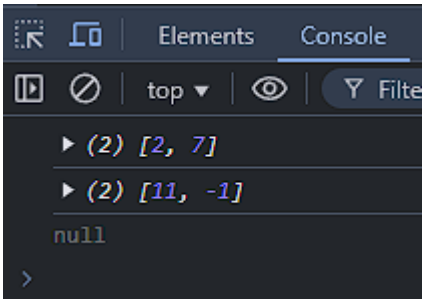


- **Array Sum:** Write an algorithm to find the pair of elements in an array that adds up to a specific target sum.

Input

```javascript
function findPairWithSum(arr, target) {
    let numSet = new Set();

    for (let num of arr) {
        let complement = target - num;
        if (numSet.has(complement)) {
            return [complement, num];
        }
        numSet.add(num);
    }

    return null; // Return null if no pair is found
}

// Example usage:
console.log(findPairWithSum([2, 7, 11, 15], 9));  // Output: [2, 7]
console.log(findPairWithSum([3, 5, -4, 8, 11, 1, -1, 6], 10)); // Output: [-1, 11]
console.log(findPairWithSum([1, 2, 3, 4], 8));    // Output: null (no pair found)
```

Output



- **Longest Substring Without Repeating Characters:** Write an algorithm to find the length of the longest substring without repeating characters in a given string.

Input

```javascript
function lengthOfLongestSubstring(s) {
    let charSet = new Set();
    let left = 0, maxLength = 0;

    for (let right = 0; right < s.length; right++) {
        while (charSet.has(s[right])) {
            charSet.delete(s[left]);
            left++;
        }

        charSet.add(s[right]);
        maxLength = Math.max(maxLength, right - left + 1);
    }

    return maxLength;
}

// Example usage:
console.log(lengthOfLongestSubstring("abcabcbb")); // Output: 3 ("abc")
console.log(lengthOfLongestSubstring("bbbbb"));    // Output: 1 ("b")
console.log(lengthOfLongestSubstring("pwwkew"));   // Output: 3 ("wke")
console.log(lengthOfLongestSubstring(""));         // Output: 0
```

Output

```
Elements   Console
top ▾   Filte
3
1
3
0
>
```