By using `DOMPurify.sanitize()`, you can ensure that any potentially harmful content is removed or escaped, reducing the risk of XSS attacks. Make sure to include the DOMPurify library in your project if you choose to use it.

Always remember to validate and sanitize any data that originates from external sources before inserting it into your DOM.

---

27 Problem solving questions:

1. Write a program to remove duplicates from an array ?

(Most Most Asked question)

```
const removeDuplicates = (array) => {
  let uniqueArr = [];

  for (let i = 0; i <= array.length - 1; i++) {
    if (uniqueArr.indexOf(array[i]) == -1) {
      uniqueArr.push(array[i]);
    }
  }

  return uniqueArr;
};

function removeDuplicates(arr) {
  // Use the Set object to remove duplicates. This works b
  return Array.from(new Set(arr));
  // return [...new Set(arr)] => another way
}

removeDuplicates([1, 2, 1, 3, 4, 2, 2, 1, 5, 6]);
```

---

2. Write a JavaScript function that takes an array of numbers and returns a new array with only the even numbers.

```
function findEvenNumbers(arr) {
    const result = [];

    for (let i = 0; i < arr.length; i++) {
        if (arr[i] % 2 === 0) {
            result.push(arr[i]); // Add even numbers to th
        }
    }

    return result;
}

// Example usage:
const numbers = [1, 2, 3, 4, 5, 6, 7, 8,-8,19, 9, 10];
console.log("Even numbers:", findEvenNumbers(numbers));
```

Time complexity: O(N)

3. How to check whether a string is palindrome or not ?

```
const checkPallindrome = (str) => {
  const len = str.length;

  for (let i = 0; i < len/2; i++) {
    if (str[i] !== str[len - i - 1]) {
      return "Not pallindrome";
    }
  }
  return "pallindrome";
};

console.log(checkPallindrome("madam"));
```

4. Find the factorial of given number ?

```
const findFactorial = (num) => {
  if (num == 0 || num == 1) {
    return 1;
  } else {
    return num * findFactorial(num - 1);
  }
};

console.log(findFactorial(4));
```

5. Program to find longest word in a given sentence ?

```
const findLongestWord = (sentence) => {
  let wordsArray = sentence.split(" ");
  let longestWord = "";

  for (let i = 0; i < wordsArray.length; i++) {
    if (wordsArray[i].length > longestWord.length) {
      longestWord = wordsArray[i];
    }
  }

  console.log(longestWord);
};

findLongestWord("Hi Iam Saikrishna Iam a UI Developer");
```

6. Write a JavaScript program to find the maximum number in an array.

```
function findMax(arr) {
    if (arr.length === 0) {
        return undefined; // Handle empty array case
    }

    let max = arr[0]; // Initialize max with the first ele

    for (let i = 1; i < arr.length; i++) {
        if (arr[i] > max) {
            max = arr[i]; // Update max if current element
        }
    }

    return max;
}

// Example usage:
const numbers = [1, 6, -33, 9, 4, 8, 2];
console.log("Maximum number is:", findMax(numbers));
```

Time complexity: O(N)

7. Write a JavaScript function to check if a given number is prime.

```
function isPrime(number) {
    if (number <= 1) {
        return false; // 1 and numbers less than 1 are not
    }

    // Loop up to the square root of the number
    for (let i = 2; i <= Math.sqrt(number); i++) {
        if (number % i === 0) {
            return false; // If divisible by any number, n
        }
    }

    return true; // If not divisible by any number, it's p
}

// Example usage:
console.log(isPrime(17)); // true
console.log(isPrime(19)); // false
```

Time complexity: O(N)

8. Program to find Reverse of a string without using built-in method ?

```
const findReverse = (sampleString) => {
  let reverse = "";

  for (let i = sampleString.length - 1; i >= 0; i--) {
    reverse += sampleString[i];
  }
  console.log(reverse);
};

findReverse("Hello Iam Saikrishna Ui Developer");
```

9. Find the smallest word in a given sentence ?

```
function findSmallestWord() {
  const sentence = "Find the smallest word";
  const words = sentence.split(' ');
  let smallestWord = words[0];

  for (let i = 1; i < words.length; i++) {
    if (words[i].length < smallestWord.length) {
        smallestWord = words[i];
    }
  }
  console.log(smallestWord);
}

findSmallestWord();
```

10. Write a function sumOfThirds(arr), which takes an array arr as an argument. This function should return a sum of every third number in the array, starting from the first one.

Directions:

If the input array is empty or contains less than 3 numbers then return 0.

The input array will contain only numbers.

```
export const sumOfThirds = (arr) => {
  let sum = 0;
  for (let i = 0; i < arr.length; i += 3) {
    sum += arr[i];
  }
  return sum;
};
```

11. Write a JavaScript function that returns the Fibonacci sequence up to a given number of terms.

```
function fibonacciSequence(numTerms) {
    if (numTerms <= 0) {
        return [];
    } else if (numTerms === 1) {
        return [0];
    }
    const sequence = [0, 1];

    for (let i = 2; i < numTerms; i++) {
        const nextFibonacci = sequence[i - 1] + sequence[i
        sequence.push(nextFibonacci);
    }

    return sequence;
}

// Example usage:
const numTerms = 10;
const fibonacciSeries = fibonacciSequence(numTerms);
console.log(fibonacciSeries); // Output: [0, 1, 1, 2, 3, 5
```

Time complexity: O(N)

12. Find the max count of consecutive 1's in an array ?

```
const findConsecutive = (array) => {
  let maxCount = 0;
  let currentConsCount = 0;

  for (let i = 0; i <= array.length - 1; i++) {
    if (array[i] === 1) {
      currentConsCount += 1;
      maxCount = Math.max(currentConsCount, maxCount);
    } else {
      currentConsCount = 0;
    }
  }

  console.log(maxCount);
};
```

```
findConsecutive([1, 1, 9, 1, 9, 9, 19, 7, 1, 1, 1, 3, 2, 5
// output: 3
```

---

13. Given 2 arrays that are sorted [0,3,4,31] and [4,6,30]. Merge them and sort [0,3,4,4,6,30,31] ?

```
const sortedData = (arr1,arr2) => {

let i = 1;
  let j=1;
  let array1 = arr1[0];
  let array2 = arr2[0];

  let mergedArray = [];

  while(array1 || array2){

    if(array2 === undefined || array1 < array2){
      mergedArray.push(array1);
      array1 = arr1[i];
      i++
    }else{
      mergedArray.push(array2);
      array2 = arr2[j];
      j++
    }

  }
  console.log(mergedArray)


}
```

```
sortedData([1,3,4,5],[2,6,8,9])
```

---

14. Create a function which will accepts two arrays arr1 and arr2. The function should return true if every value in arr1 has its corresponding value squared in array2. The frequency of values must be same. (Effecient)

Inputs and outputs:

==============

[1,2,3],[4,1,9] ⇒ true

[1,2,3],[1,9] ==⇒ false

[1,2,1],[4,4,1] =⇒ false (must be same frequency)

```
function isSameFrequency(arr1,arr2){

  if(arr1.length !== arr2.length){
     return false;
  }

  let arrFreq1={};
  let arrFreq2={};

  for(let val of arr1){
    arrFreq1[val] = (arrFreq1[val] || 0) + 1;
  }

  for(let val of arr2){
    arrFreq2[val] = (arrFreq2[val] || 0) + 1;
  }

  for(let key in arrFreq1){
        if(!key*key in arrFreq2) return false;
        if(arrFreq1[key] !== arrFreq2[key*key]) return fal
  }
  return true;

}

console.log(isSameFrequency([1,2,5],[25,4,1]))
```

---

15. Given two strings. Find if one string can be formed by rearranging the letters of other string. (Effecient)

Inputs and outputs:

"aaz","zza" ⇒ false

"qwerty","qeywrt" ⇒ true

```
function isStringCreated(str1,str2){
  if(str1.length !== str2.length) return false
  let freq = {};

  for(let val of str1){
    freq[val] = (freq[val] || 0) + 1;
  }

  for(let val of str2){
    if(freq[val]){
      freq[val] -= 1;
    } else{
      return false;
    }
  }
  return true;
}
```

```
console.log(isStringCreated('anagram','nagaram'))
```

16. Write logic to get unique objects from below array ?

I/P: [{name: "sai"},{name:"Nang"},{name: "sai"},{name:"Nang"},{name: "111111"}];

O/P: [{name: "sai"},{name:"Nang"}{name: "111111"}

```
function getUniqueArr(array){
    const uniqueArr = [];
    const seen = {};
    for(let i=0; i<=array.length-1;i++){
        const currentItem = array[i].name;
        if(!seen[currentItem]){
            uniqueArr.push(array[i]);
            seen[currentItem] = true;
        }
    }
    return uniqueArr;
}

let arr = [{name: "sai"},{name:"Nang"},{name: "sai"},{name
console.log(getUniqueArr(arr))
```

17. Write a JavaScript program to find the largest element in a nested array.

```
function findLargestElement(arr) {
    let max = Number.NEGATIVE_INFINITY; // Initialize max

    // Helper function to traverse nested arrays
    function traverse(arr) {
        for (let i = 0; i < arr.length; i++) {
            if (Array.isArray(arr[i])) {
                // If element is an array, recursively cal
                traverse(arr[i]);
            } else {
                // If element is not an array, update max
                if (arr[i] > max) {
                    max = arr[i];
                }
            }
        }
    }

    // Start traversing the input array
    traverse(arr);

    return max;
}

// Example usage:
const nestedArray = [[3, 4, 58], [709, 8, 9, [10, 11]], [1
console.log("Largest element:", findLargestElement(nestedA
```

Time complexity: O(N)

---

18. Given a string, write a javascript function to count the occurrences of each character in the string.

```
function countCharacters(str) {
    const charCount = {}; // Object to store character cou
    const len = str.length;

    // Loop through the string and count occurrences of ea
    for (let i = 0; i < len; i++) {
        const char = str[i];
        // Increment count for each character
        charCount[char] = (charCount[char] || 0) + 1;
    }

    return charCount;
}

// Example usage:
const result = countCharacters("helaalo");
console.log(result); // Output: { h: 1, e: 1, l: 2, o: 1 }
```

Time complexity: O(N)

19. Write a javascript function that sorts an array of numbers in ascending order.

```
function quickSort(arr) {
    // Check if the array is empty or has only one element
    if (arr.length <= 1) {
        return arr;
    }

    // Select a pivot element
    const pivot = arr[0];

    // Divide the array into two partitions
    const left = [];
    const right = [];

    for (let i = 1; i < arr.length; i++) {
        if (arr[i] < pivot) {
            left.push(arr[i]);
        } else {
            right.push(arr[i]);
        }
    }
    // Recursively sort the partitions
    const sortedLeft = quickSort(left);
    const sortedRight = quickSort(right);

    // Concatenate the sorted partitions with the pivot an
    return sortedLeft.concat(pivot, sortedRight);
}

// Example usage:
const unsortedArray = [5, 2, 9, 1, 3, 6];
const sortedArray = quickSort(unsortedArray);
console.log(sortedArray); // Output: [1, 2, 3, 5, 6, 9]
```

Time complexity: O(n log n)

20. Write a javascript function that sorts an array of numbers in descending order.

```
function quickSort(arr) {
    if (arr.length <= 1) {
        return arr;
    }

    const pivot = arr[0];
    const left = [];
    const right = [];

    for (let i = 1; i < arr.length; i++) {
        if (arr[i] >= pivot) {
            left.push(arr[i]);
        } else {
            right.push(arr[i]);
        }
    }

    return [...quickSort(left), pivot, ...quickSort(right)
```

```
}

const arr = [3, 1, 4, 1, 5, 9, 2, 6, 5];
const sortedArr = quickSort(arr);
console.log(sortedArr); // Output: [9, 6, 5, 5, 4, 3, 2, 1
```

Time complexity: O(n log n)

21. Write a javascript function that reverses the order of words in a sentence without using the built-in reverse() method.

```
const reverseWords = (sampleString) => {
  let reversedSentence = "";
  let word = "";

  // Iterate over each character in the sampleString
  for (let i = 0; i < sampleString.length; i++) {
    // If the character is not a space, append it to the c
    if (sampleString[i] !== ' ') {
      word += sampleString[i];
    } else {
      // If it's a space, prepend the current word to the
      //reset the word
      reversedSentence = word + ' ' + reversedSentence;
      word = "";
    }
  }

  // Append the last word to the reversed sentence
  reversedSentence = word + ' ' + reversedSentence;

  // Trim any leading or trailing spaces and log the resul
  console.log(reversedSentence.trim());
};
```

```
// Example usage
reverseWords("ChatGPT is awesome"); //"awesome is ChatGPT"
```

```
function reverseWords(sentence) {
    // Split the sentence into words
    let words = [];
    let wordStart = 0;
    for (let i = 0; i < sentence.length; i++) {
        if (sentence[i] === ' ') {
            words.unshift(sentence.substring(wordStart, i)
            wordStart = i + 1;
        } else if (i === sentence.length - 1) {
            words.unshift(sentence.substring(wordStart, i
        }
    }

    // Join the words to form the reversed sentence
    return words.join(' ');
}

// Example usage
const sentence = "ChatGPT is awesome";
console.log(reverseWords(sentence)); // Output: "awesome i
```

Time complexity: O(N)

22. Implement a javascript function that flattens a nested array into a single-dimensional array.

```
function flattenArray(arr) {
    const stack = [...arr];
    const result = [];

    while (stack.length) {
        const next = stack.pop();
        if (Array.isArray(next)) {
                stack.push(...next);
            } else {
                result.push(next);
            }
        }

    return result.reverse(); // Reverse the result to main
}

// Example usage:
const nestedArray = [1, [2, [3, 4], [7,5]], 6];
const flattenedArray = flattenArray(nestedArray);
console.log(flattenedArray); // Output: [1, 2, 3, 4, 5, 6]
```

---

23. Write a function which converts string input into an object

// stringToObject("a.b.c", "someValue");

// output → {a: {b: {c: "someValue"}}}

```
function stringToObject(str, finalValue) {
  const keys = str.split('.');
  let result = {};
  let current = result;

  for (let i = 0; i < keys.length; i++) {
    const key = keys[i];
    current[key] = (i === keys.length - 1) ? finalValue :
    current = current[key];
  }

  return result;
}

// Test the function
```

```
const output = stringToObject("a.b.c", "someValue");
console.log(output); // Output: {a: {b: {c: "someValue"}}}
```

---

24. Given an array, return an array where the each value is the product of the next two items: E.g. `[3, 4, 5]` -> `[20, 15, 12]`

```
function productOfNextTwo(arr) {
    const result = [];
    for (let i = 0; i < arr.length; i++) {
        if (i < arr.length - 1) {
            result.push(arr[i + 1] * arr[i + 2]);
        } else {
            result.push(arr[0] * arr[1]);
        }
    }
    return result;
}

// Example usage:
const inputArray = [3, 4, 5];
const outputArray = productOfNextTwo(inputArray);
console.log(outputArray); // Output: [20, 15, 12]
```

---

25. Find the 2nd largest element from a given array ? [100,20,112,22]

```
function findSecondLargest(arr) {
    if (arr.length < 2) {
        throw new Error("Array must contain at least two e

    }

    let largest = -Infinity;
    let secondLargest = -Infinity;

    for (let i = 0; i < arr.length; i++) {
        if (arr[i] > largest) {
            secondLargest = largest;
            largest = arr[i];
        } else if (arr[i] > secondLargest && arr[i] < larg
            secondLargest = arr[i];
        }
    }

    if (secondLargest === -Infinity) {
        throw new Error("There is no second largest elemen
    }

    return secondLargest;
}

// Example usage:
const array = [10, 5, 20, 8, 12];
console.log(findSecondLargest(array)); // Output: 12
```

26. Program challenge: Find the pairs from given input ?

input1 = [1, 2, 3, 4, 5, 6, 7, 8, 9];

input2 = 10;

output = [[4, 6], [3, 7], [2, 8], [1, 9]]

```
function findPairs(input1, input2) {
  const pairs = [];
  const seen = new Set();

  for (const num of input1) {
    const complement = input2 - num;
    if (seen.has(complement)) {
      pairs.push([complement, num]);
    }
    seen.add(num);
  }

  return pairs;
}

const input1 = [1, 2, 3, 4, 5, 6, 7, 8, 9];
const input2 = 10;

const output = findPairs(input1, input2);
console.log(output); // [[1, 9], [2, 8], [3, 7], [4, 6], [
```

---

27. Write a javascript program to get below output from given input ?

I/P: abbcccddddeea

O/P: 1a2b3c4d2e1a

```
function encodeString(input) {
    if (input.length === 0) return "";

    let result = "";
    let count = 1;

    for (let i = 1; i < input.length; i++) {
        if (input[i] === input[i - 1]) {
            count++;
        } else {
            result += count + input[i - 1];
            count = 1;
        }
    }
    // Add the last sequence
    result += count + input[input.length - 1];

    return result;
}

const input = "abbcccddddeea";
const output = encodeString(input);
console.log(output);  // Outputs: 1a2b3c4d2e1a
```

---

52 Reactjs Interview questions & Answers

1. What is React?

- React is an opensource component based JavaScript library which is used to develop interactive user interfaces.

---

2. What are the features of React ?

- Jsx
- Virtual dom
- one way data binding
- Uses reusable components to develop the views
- Supports server side rendering

---

3. What is JSX ?