Coding question-4

Interviewquestion-51

JS

Counting Occurrences

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

Output : { 1: 1, 2: 2, 3: 3, 4: 4 }

Few approaches are mentioned in the following slides

Approach 1: Normal approach

```javascript
function countOccurrences(array) {
  const counts = {}; // Create an empty object to store counts

  // Iterate through the array
  for (const number of array) {
    if (counts[number]) {
      // If the number is already in the counts object, increment its count
      counts[number]++;
    } else {
      // If the number is not in the counts object, initialize its count to
      counts[number] = 1;
    }
  }

  return counts;
}

const numbers = [1, 2, 2, 3, 3, 3, 4, 4, 4, 4];
const counts = countOccurrences(numbers);

console.log(counts);
```

Approach 2: Best Approach

```javascript
const numbers = [1, 2, 3, 2, 4, 1, 5, 3, 1];
const numberCounts = numbers.reduce((counts, number) => {
  counts[number] = (counts[number] || 0) + 1;
  return counts;
}, {});
console.log(numberCounts);
```

Explanation

- **counts** is an object that stores the counts of each unique number.
- **number** is the current number being processed in the array.

- **(counts[number] || 0):** This part retrieves the current count of number from the counts object. If number doesn't exist as a property in counts, it returns 0. This is achieved using the logical OR (||) operator. It's a way to handle the case when number is encountered for the first time, and there is no existing count.

- **+ 1:** This part increments the count obtained in step 1 by 1. It adds 1 to the current count to update the count for the current number.

- **counts[number] = ...:** Finally, this part assigns the updated count back to the counts object, associating it with the number as the property name. If number didn't exist before, this creates a new property in the counts object.

In this way we can count the duplicates of each value