JavaScript Beg-Inter Questions

Beginner Level Questions

1. Create an array and print its elements.

Question: Ek array banao aur uske elements print karo.

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

Explanation: Yah code ek array create karta hai aur console me uske elements ko print karta hai.

2. Find the length of an array.

Question: Array ki length find karo.

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

Explanation: `length` property use karke array ki total length ya elements ki sankhya nikaali ja sakti hai.

3. Access the first element of an array.

Question: Array ke pehle element ko access karo.

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

Explanation: Array ke first element ko access karne ke liye index 0 use hota hai.

4. Access the last element of an array.

Question: Array ke last element ko access karo.

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

Explanation: Last element ko access karne ke liye, array length se 1 minus karte hain.

5. Add an element at the end of an array.

Question: Array ke end me ek element add karo.

```
let arr = [1, 2, 3, 4, 5];
arr.push(6);
console.log(arr);
```

Explanation: `push` method use karke hum array ke end me new element add kar sakte hain.

6. Remove the last element from an array.

Question: Array ke last element ko remove karo.

```
let arr = [1, 2, 3, 4, 5];
arr.pop();
console.log(arr);
```

Explanation: `pop` method use karke array ke last element ko remove karte hain.

7. Add an element at the beginning of an array.

Question: Array ke beginning me ek element add karo.

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

Explanation: `unshift` method array ke beginning me new element add karne ke liye use hota hai.

8. Remove the first element from an array.

Question: Array ke first element ko remove karo.

```
let arr = [1, 2, 3, 4, 5];
arr.shift();
console.log(arr);
```

Explanation: `shift` method array ke first element ko remove karta hai.

9. Concatenate two arrays.

Question: Do arrays ko concatenate (milana) karo.

```
let arr1 = [1, 2, 3];
let arr2 = [4, 5, 6];
let combined = arr1.concat(arr2);
console.log(combined);
```

Explanation: `concat` method use karke do arrays ko combine kar sakte hain.

10. Check if an array contains a specific element.

Question: Check karo ki array me specific element hai ya nahi.

```
let arr = [1, 2, 3, 4, 5];
console.log(arr.includes(3));
```

Explanation: `includes` method check karta hai ki given element array me hai ya nahi.

Intermediate Level Questions

11. Reverse an array.

Question: Array ko reverse karo.

```
let arr = [1, 2, 3, 4, 5];
arr.reverse();
```

```
console.log(arr);
```

Explanation: `reverse` method array ke elements ko reverse kar deta hai.

12. Sort an array in ascending order.

Question: Array ko ascending order me sort karo.

```
let arr = [3, 1, 4, 1, 5, 9];
arr.sort((a, b) => a - b);
console.log(arr);
```

Explanation: `sort` method aur comparison function use karke array ko ascending order me sort karte hain.

13. Sort an array in descending order.

Question: Array ko descending order me sort karo.

```
let arr = [3, 1, 4, 1, 5, 9];
arr.sort((a, b) => b - a);
console.log(arr);
```

Explanation: `sort` method aur comparison function use karke array ko descending order me sort karte hain.

14. Find the index of a specific element.

Question: Specific element ka index find karo.

```
let arr = [1, 2, 3, 4, 5];
console.log(arr.indexOf(3));
```

Explanation: `indexOf` method use karke kisi element ka index nikaalte hain.

15. Remove a specific element by its index.

Question: Specific element ko uske index se remove karo.

```
let arr = [1, 2, 3, 4, 5];
arr.splice(arr.indexOf(3), 1);
console.log(arr);
```

Explanation: `splice` method use karke kisi element ko uske index se remove karte hain.

16. Create a new array with only even numbers.

Question: Naya array banao jisme sirf even numbers ho.

```
let arr = [1, 2, 3, 4, 5, 6];
let evens = arr.filter(num => num % 2 === 0);
console.log(evens);
```

Explanation: `filter` method use karke array se even numbers ko filter karte hain.

17. Sum all elements of an array.

Question: Array ke saare elements ka sum find karo.

```
let arr = [1, 2, 3, 4, 5];
let sum = arr.reduce((total, num) => total + num, 0);
console.log(sum);
```

Explanation: `reduce` method use karke array ke saare elements ka sum nikaalte hain.

18. Find the maximum element in an array.

Question: Array me maximum element find karo.

```
let arr = [1, 2, 3, 4, 5];
let max = Math.max(...arr);
console.log(max);
```

Explanation: `Math.max` function aur spread operator use karke array me max element find karte hain.

19. Find the minimum element in an array.

Question: Array me minimum element find karo.

```
let arr = [1, 2, 3, 4, 5];
let min = Math.min(...arr);
console.log(min);
```

Explanation: `Math.min` function aur spread operator use karke array me min element find karte hain.

20. Flatten a nested array.

Question: Nested array ko flatten karo.

```
let arr = [1, [2, 3], [4, [5, 6]]];
let flat = arr.flat(2);
console.log(flat);
```

Explanation: `flat` method use karke nested array ko flatten karte hain.

21. Remove duplicate elements from an array.

Question: Array se duplicate elements ko remove karo.

```
let arr = [1, 2, 3, 3, 4, 4, 5];
let unique = [...new Set(arr)];
console.log(unique);
```

Explanation: `Set` aur spread operator use karke array se duplicates remove karte hain.

22. Merge two arrays and remove duplicates.

Question: Do arrays ko merge karo aur duplicates remove karo.

```
let arr1 = [1, 2, 3];
let arr2 = [3, 4, 5];
let merged = [...new Set([...arr1, ...arr2])];
console.log(merged);
```

Explanation: `Set` aur spread operator use karke do arrays ko merge karte hain aur duplicates remove karte hain.

23. Create an array of the first n Fibonacci numbers.

Question: Pehle n Fibonacci numbers ka array banao.

```
function fibonacci(n) {
    let fib = [0, 1];
    for (let i = 2; i < n; i++) {
        fib[i] = fib[i - 1] + fib[i - 2];
    }
    return fib;
}
console.log(fibonacci(10));
```

Explanation: Yah function pehle n Fibonacci numbers ko generate karke array me return karta hai.

24. Create an array of squares of each element.

Question: Har element ka square nikal ke array banao.

```
let arr = [1, 2, 3, 4, 5];
let squares = arr.map(num => num ** 2);
console.log(squares);
```

Explanation: `map` method use karke har element ka square nikaal ke new array banate hain.

25. Find all elements greater than a specific value.

Question: Specific value se greater saare elements find karo.

```
let arr = [1, 2, 3, 4, 5];
let greaterThanThree = arr.filter(num => num > 3);
console.log(greaterThanThree);
```

Explanation: `filter` method use karke specific value se greater elements ko filter karte hain.

26. Check if all elements are positive.

Question: Check karo ki saare elements positive hain ya nahi.

```
let arr = [1, 2, 3, 4, 5];
let allPositive = arr.every(num => num > 0);
console.log(allPositive);
```

Explanation: `every` method use karke check karte hain ki saare elements positive hain ya nahi.

27. Find the difference between the largest and smallest numbers.

Question: Largest aur smallest number ke beech ka difference find karo.

```
let arr = [1, 2, 3, 4, 5];
let max = Math.max(...arr);
let min = Math.min(...arr);
console.log(max - min);
```

Explanation: `Math.max` aur `Math.min` functions use karke difference nikaalte hain.

28. Rotate an array by k positions.

Question: Array ko k positions se rotate karo.

```
function rotate(arr, k) {
    k = k % arr.length;
    return [...arr.slice(k), ...arr.slice(0, k)];
}
let arr = [1, 2, 3, 4, 5];
console.log(rotate(arr, 2));
```

Explanation: `slice` aur `spread` operator use karke array ko k positions se rotate karte hain.

29. Find the second largest element in an array.

Question: Array me second largest element find karo.

```
let arr = [1, 2, 3, 4, 5];
let max = Math.max(...arr);
let filteredArr = arr.filter(num => num !== max);
let secondMax = Math.max(...filteredArr);
console.log(secondMax);
```

Explanation: Pehle largest element remove karte hain aur fir bacha hua max element nikaalte hain.

30. Implement a function to find the intersection of two arrays.

Question: Do arrays ke intersection elements find karne ka function banao.

```
function intersection(arr1, arr2) {
    return arr1.filter(value => arr2.includes(value));
}
let arr1 = [1, 2, 3, 4, 5];
let arr2 = [3, 4, 5, 6, 7];
console.log(intersection(arr1, arr2));
```

Explanation: `filter` aur `includes` method use karke do arrays ke common elements find karte hain.

Advanced Level Questions

31. Find the union of two arrays.

Question: Do arrays ka union find karo.

```
function union(arr1, arr2) {
    return [...new Set([...arr1, ...arr2])];
}
let arr1 = [1, 2, 3, 4];
let arr2 = [3, 4, 5, 6];
console.log(union(arr1, arr2));
```

Explanation: `Set` aur spread operator use karke do arrays ka union banate hain.

32. Find the symmetric difference of two arrays.

Question: Do arrays ka symmetric difference find karo.

```
function symmetricDifference(arr1, arr2) {
    return [...arr1.filter(x => !arr2.includes(x)), ...arr2.f
ilter(x => !arr1.includes(x))];
}
let arr1 = [1, 2, 3];
let arr2 = [2, 3, 4];
console.log(symmetricDifference(arr1, arr2));
```

Explanation: `filter` aur `includes` method use karke symmetric difference nikaalte hain.

33. Implement binary search on a sorted array.

Question: Sorted array par binary search implement karo.

```
function binarySearch(arr, target) {
    let left = 0, right = arr.length - 1;
    while (left <= right) {
        let mid = Math.floor((left + right) / 2);
```

```
        if (arr[mid] === target) return mid;
        else if (arr[mid] < target) left = mid + 1;
        else right = mid - 1;
    }
    return -1;
}
let arr = [1, 2, 3, 4, 5];
console.log(binarySearch(arr, 3));
```

Explanation: Binary search algorithm use karke sorted array me target element find karte hain.

34. Find the first duplicate element in an array.

Question: Array me pehla duplicate element find karo.

```
function firstDuplicate(arr) {
    let seen = new Set();
    for (let num of arr) {
        if (seen.has(num)) return num;
        seen.add(num);
    }
    return -1;
}
let arr = [1, 2, 3, 4, 2, 5];
console.log(firstDuplicate(arr));
```

Explanation: `Set` use karke pehla duplicate element find karte hain.

35. Group elements of an array based on their frequency.

Question: Array ke elements ko unki frequency ke base par group karo.

```
function groupByFrequency(arr) {
    let frequencyMap = arr.reduce((acc, val) => {
        acc[val] = (acc[val] || 0) + 1;
        return acc;
```

```
    }, {});
    return Object.keys(frequencyMap).map(key => ({
        element: key,
        frequency: frequencyMap[key]
    }));
}
let arr = [1, 2, 2, 3, 3, 3];
console.log(groupByFrequency(arr));
```

Explanation: `reduce` aur `map` methods use karke elements ko unki frequency ke base par group karte hain.

36. Find the subarray with the maximum sum (Kadane's Algorithm).

Question: Maximum sum wala subarray find karo (Kadane's Algorithm).

```
function maxSubArraySum(arr) {
    let maxSoFar = -Infinity, maxEndingHere = 0;
    for (let num of arr) {
        maxEndingHere = Math.max(num, maxEndingHere + num);
        maxSoFar = Math.max(maxSoFar, maxEndingHere);
    }
    return maxSoFar;
}
let arr = [-2, 1, -3, 4, -1, 2, 1, -5, 4];
console.log(maxSubArraySum(arr));
```

Explanation: Kadane's Algorithm use karke maximum sum subarray find karte hain.

37. Find the longest increasing subsequence.

Question: Longest increasing subsequence find karo.

```
function lengthOfLIS(arr) {
    let dp = Array(arr.length).fill(1);
```

```
    for (let i = 1; i < arr.length; i++) {
        for (let j = 0; j < i; j++) {
            if (arr[i] > arr[j]) {
                dp[i] = Math.max(dp[i], dp[j] + 1);
            }
        }
    }
    return Math.max(...dp);
}
let arr = [10, 9, 2, 5, 3, 7, 101, 18];
console.log(lengthOfLIS(arr));
```

Explanation: Dynamic programming approach use karke longest increasing subsequence find karte hain.

38. Rotate an array to the right by k steps.

Question: Array ko right side se k steps se rotate karo.

```
function rotateRight(arr, k) {
    k = k % arr.length;
    return [...arr.slice(-k), ...arr.slice(0, -k)];
}
let arr = [1, 2, 3, 4, 5];
console.log(rotateRight(arr, 2));
```

Explanation: `slice` aur `spread` operator use karke array ko right side se k steps rotate karte hain.

39. Find the majority element (element appearing more than n/2 times).

Question: Majority element find karo (jo n/2 times se zyada appear hota hai).

```
function majorityElement(arr) {
    let count = 0, candidate = null;
    for (let num of arr) {
```

```
        if (count === 0) candidate = num;
        count += (num === candidate) ? 1 : -1;
    }
    return candidate;
}
let arr = [3, 2, 3];
console.log(majorityElement(arr));
```

Explanation: Boyer-Moore Voting Algorithm use karke majority element find karte hain.

40. Find the missing number in an array of 1 to n.

Question: 1 se n tak ke array me missing number find karo.

```
function missingNumber(arr) {
    let n = arr.length + 1;
    let totalSum = (n * (n + 1)) / 2;
    let arrSum = arr.reduce((a, b) => a + b, 0);
    return totalSum - arrSum;
}
let arr = [1, 2, 4, 5, 6];
console.log(missingNumber(arr));
```

Explanation: Total sum of 1 to n aur array sum ke difference se missing number find karte hain.

41. Find the peak element in an array.

Question: Array me peak element find karo.

```
function findPeakElement(arr) {
    for (let i = 0; i < arr.length; i++) {
        if ((i === 0 || arr[i] > arr[i - 1]) && (i === arr.le
ngth - 1 || arr[i] > arr[i + 1])) {
            return arr[i];
        }
```

```
    }
}
let arr = [1, 2, 3, 1];
console.log(findPeakElement(arr));
```

Explanation: Linear search use karke peak element find karte hain jo apne dono neighbors se bada hota hai.

42. Find the smallest missing positive number.

Question: Smallest missing positive number find karo.

```
function firstMissingPositive(arr) {
    let set = new Set(arr);
    for (let i = 1; i <= arr.length; i++) {
        if (!set.has(i)) return i;
    }
    return arr.length + 1;
}
let arr = [3, 4, -1, 1];
console.log(firstMissingPositive(arr));
```

Explanation: `Set` use karke smallest missing positive number find karte hain.

43. Find the maximum product of two integers in an array.

Question: Array me do integers ka maximum product find karo.

```
function maxProductOfTwo(arr) {
    let max1 = Math.max(...arr);
    arr.splice(arr.indexOf(max1), 1);
    let max2 = Math.max(...arr);
    return max1 * max2;
}
let arr = [1, 2, 3, 4, 5];
console.log(maxProductOfTwo(arr));
```

Explanation: Pehle do largest numbers find karte hain aur unka product nikaalte hain.

44. Implement the quicksort algorithm.

Question: Quicksort algorithm implement karo.

```
function quicksort(arr) {
    if (arr.length <= 1) return arr;
    let pivot = arr[Math.floor(arr.length / 2)];
    let left = arr.filter(x => x < pivot);
    let middle = arr.filter(x => x === pivot);
    let right = arr.filter(x => x > pivot);
    return [...quicksort(left), ...middle, ...quicksort(righ
t)];
}
let arr = [3, 6, 8, 10, 1, 2, 1];
console.log(quicksort(arr));
```

Explanation: Quicksort algorithm use karke array ko sort karte hain.

45. Find the largest sum of contiguous subarray (Kadane's Algorithm).

Question: Largest

sum of contiguous subarray find karo (Kadane's Algorithm).

```
function maxSubArraySum(arr) {
    let maxSoFar = arr[0], maxEndingHere = arr[0];
    for (let i = 1; i < arr.length; i++) {
        maxEndingHere = Math.max(arr[i], maxEndingHere + arr
[i]);
        maxSoFar = Math.max(maxSoFar, maxEndingHere);
    }
    return maxSoFar;
}
```

```
let arr = [-2, 1, -3, 4, -1, 2, 1, -5, 4];
console.log(maxSubArraySum(arr));
```

Explanation: Kadane's Algorithm use karke largest sum of contiguous subarray find karte hain.

46. Implement merge sort algorithm.

Question: Merge sort algorithm implement karo.

```
function mergeSort(arr) {
    if (arr.length <= 1) return arr;
    let mid = Math.floor(arr.length / 2);
    let left = mergeSort(arr.slice(0, mid));
    let right = mergeSort(arr.slice(mid));
    return merge(left, right);
}

function merge(left, right) {
    let result = [], leftIndex = 0, rightIndex = 0;
    while (leftIndex < left.length && rightIndex < right.leng
th) {
        if (left[leftIndex] < right[rightIndex]) {
            result.push(left[leftIndex]);
            leftIndex++;
        } else {
            result.push(right[rightIndex]);
            rightIndex++;
        }
    }
    return result.concat(left.slice(leftIndex)).concat(right.
slice(rightIndex));
}
let arr = [3, 6, 8, 10, 1, 2, 1];
console.log(mergeSort(arr));
```

Explanation: Merge sort algorithm use karke array ko sort karte hain.

47. Find the longest common subsequence of two arrays.

Question: Do arrays ka longest common subsequence find karo.

```
function longestCommonSubsequence(arr1, arr2) {
    let dp = Array.from({ length: arr1.length + 1 }, () => Ar
ray(arr2.length + 1).fill(0));
    for (let i = 1; i <= arr1.length; i++) {
        for (let j = 1; j <= arr2.length; j++) {
            if (arr1[i - 1] === arr2[j - 1]) {
                dp[i][j] = dp[i - 1][j - 1] + 1;
            } else {
                dp[i][j] = Math.max(dp[i - 1][j], dp[i][j -
1]);
            }
        }
    }
    return dp[arr1.length][arr2.length];
}
let arr1 = [1, 3, 4, 1];
let arr2 = [3, 4, 1, 2, 1, 3];
console.log(longestCommonSubsequence(arr1, arr2));
```

Explanation: Dynamic programming use karke do arrays ka longest common subsequence find karte hain.

48. Implement the radix sort algorithm.

Question: Radix sort algorithm implement karo.

```
function radixSort(arr) {
    const getMax = arr => Math.max(...arr);
    const getDigit = (num, place) => Math.floor(Math.abs(num)
/ Math.pow(10, place)) % 10;
    const digitCount = num => (num === 0 ? 1 : Math.floor(Mat
```

```
h.log10(Math.abs(num))) + 1);
    const mostDigits = arr => Math.max(...arr.map(digitCoun
t));

    let maxDigitCount = mostDigits(arr);
    for (let k = 0; k < maxDigitCount; k++) {
        let digitBuckets = Array.from({ length: 10 }, () =>
[]);
        for (let i = 0; i < arr.length; i++) {
            let digit = getDigit(arr[i], k);
            digitBuckets[digit].push(arr[i]);
        }
        arr = [].concat(...digitBuckets);
    }
    return arr;
}
let arr = [23, 345, 5467, 12, 2345, 9852];
console.log(radixSort(arr));
```

Explanation: Radix sort algorithm use karke array ko sort karte hain.

49. Find the kth smallest element in an unsorted array.

Question: Unsorted array me kth smallest element find karo.

```
function kthSmallest(arr, k) {
    arr.sort((a, b) => a - b);
    return arr[k - 1];
}
let arr = [7, 10, 4, 3, 20, 15];
console.log(kthSmallest(arr, 3));
```

Explanation: Array ko sort karke kth smallest element ko return karte hain.

50. Find the kth largest element in an unsorted array.

Question: Unsorted array me kth largest element find karo.

```
function kthLargest(arr, k) {
    arr.sort((a, b) => b - a);
    return arr[k - 1];
}
let arr = [7, 10, 4, 3, 20, 15];
console.log(kthLargest(arr, 3));
```

Explanation: Array ko sort karke kth largest element ko return karte hain.

51. Implement heap sort algorithm.

Question: Heap sort algorithm implement karo.

```
function heapSort(arr) {
    const heapify = (arr, length, i) => {
        let largest = i;
        let left = 2 * i + 1;
        let right = 2 * i + 2;
        if (left < length && arr[left] > arr[largest]) {
            largest = left;
        }
        if (right < length && arr[right] > arr[largest]) {
            largest = right;
        }
        if (largest !== i) {
            [arr[i], arr[largest]] = [arr[largest], arr[i]];
            heapify(arr, length, largest);
        }
    };

    let length = arr.length;
    for (let i = Math.floor(length / 2) - 1; i >= 0; i--) {
        heapify(arr, length, i);
    }

    for (let i = length - 1; i > 0; i--) {
```

```
            [arr[0], arr[i]] = [arr[i], arr[0]];
            heapify(arr, i, 0);
        }
        return arr;
}
let arr = [12, 11, 13, 5, 6, 7];
console.log(heapSort(arr));
```

Explanation: Heap sort algorithm use karke array ko sort karte hain.

52. Find the length of the longest subarray with sum zero.

Question: Sum zero wala longest subarray ka length find karo.

```
function maxLenZeroSumSubarray(arr) {
    let sumMap = new Map();
    let maxLen = 0, sum = 0;
    for (let i = 0; i < arr.length; i++) {
        sum += arr[i];
        if (sum === 0) {
            maxLen = i + 1;
        } else if (sumMap.has(sum)) {
            maxLen = Math.max(maxLen, i - sumMap.get(sum));
        } else {
            sumMap.set(sum, i);
        }
    }
    return maxLen;
}
let arr = [15, -2, 2, -8, 1, 7, 10, 23];
console.log(maxLenZeroSumSubarray(arr));
```

Explanation: Hashmap use karke sum zero wala longest subarray ka length find karte hain.

53. Find the longest palindrome subarray.

Question: Longest palindrome subarray find karo.

```
function isPalindrome(arr, start, end) {
    while (start < end) {
        if (arr[start] !== arr[end]) return false;
        start++;
        end--;
    }
    return true;
}

function longestPalindromeSubarray(arr) {
    let maxLength = 1, start = 0;
    for (let i = 0; i < arr.length; i++) {
        for (let j = i; j < arr.length; j++) {
            if (isPalindrome(arr, i, j) && (j - i + 1) > maxL
ength) {
                start = i;
                maxLength = j - i + 1;
            }
        }
    }
    return arr.slice(start, start + maxLength);
}
let arr = [1, 2, 3, 4, 3, 2, 1];
console.log(longestPalindromeSubarray(arr));
```

Explanation: Nested loops aur helper function use karke longest palindrome subarray find karte hain.