1.Given an array of n integers. The task is to print the duplicates in the given array. If there are no duplicates then print -1.
Examples:
Input: {2, 10,10, 100, 2, 10, 11,2,11,2}
Output: 2 10 11

Input: {5, 40, 1, 40, 100000, 1, 5, 1}
Output: 5 40 1

Answer:

```
function findDuplicates(a) {
   const count = {};
   for (const num of a) {
      if(count[num]){
         count[num]++
      }
      else{
         count[num]=1
      }
   }
   const duplicates = Object.keys(count).filter(num => count[num] > 1);
   console.log(duplicates.length>0?duplicates.join(' '):-1);
}
findDuplicates([1,2,3,4,5,1,2,3]);
```

2. Given an array of size N. The task is to find the maximum and the minimum element of the array using the minimum number of comparisons.
Examples:
Input: arr[] = {3, 5, 4, 1, 9}

Output: Minimum element is: 1 and Maximum element is: 9

Answer:

```
const a=[1,2,3,6,8,4,7,10]
const max=a.reduce((a,b)=>a>b?a:b)
const min=a.reduce((a,b)=>a<b?a:b)
console.log(`Maximum element=${max} and Minimum element=${min}`);
```

3. Sorting an array means arranging the elements of the array in a certain order. Generally sorting in an array is done to arrange the elements in increasing or decreasing order.
Problem statement: Given an array of integers arr, the task is to sort the array in ascending order and

return it, without using any built-in functions.
Example:
Input: arr = [5, 2, 4, 3, 1] Output: [1, 2, 3, 4, 5]

Answer:

```
function sort(a){
   const l=a.length
   for(var i=0;i<l-1;i++){
      let min=i;
      for(var j=i+1;j<l;j++){
         if(a[j]<a[min]){
            min=j
         }
      }
      if(min!=i){
         let temp=a[i];
         a[i]=a[min];
         a[min]=temp
      }
   }
   return a
}
console.log(sort([3,1,6,2,5]));
```

4. Given an array arr[] of size N-1 with integers in the range of [1, N], the task is to find the missing number from the first N integers.
Note: There are no duplicates in the list.
Examples:
Input: arr[] = {1, 2, 4, 6, 3, 7, 8} , N = 8

Output: 5

Explanation: Here the size of the array is 8, so the range will be [1, 8]. The missing number between 1 to 8 is 5

Answer:

```
function missingElement(a,n){
   a.sort((a,b)=>a-b)
   var low=0;
   var high=n-1;
   while(low<=high){
      var mid=Math.floor((low+high)/2)
      if(a[mid]==mid+1){
         low=mid+1
      }
```

```
      else{
        high=mid-1
      }
    }
    return low+1
}
console.log(missingElement([1,3,5,4,7,6],7));
```

5 . Write a JavaScript program to find the most frequent item in an array.
Sample array : var arr1=[3, 'a', 'a', 'a', 2, 3, 'a', 3, 'a', 2, 4, 9, 3];
Sample Output : a (5 times)

Answer:

```
function frequentItem(a){
    const items={}
    let mostItem=null
    let mostCount=0
    for(const i of a){
      if(items[i]){
        items[i]++
        if (items[i] > mostCount) {
          mostCount = items[i];
          mostItem = i;
        }
      }
      else{
        items[i]=1
      }

    };
    console.log(items);
    console.log(`${mostItem} (${mostCount} times)`);
}
frequentItem([3,'a',4,'a',4,'a'])
```

6. There are two arrays with individual values. Write a JavaScript program to compute the sum of each individual index value in the given array.
Sample array :
array1 = [1,0,2,3,4];
array2 = [3,5,6,7,8,13];
Expected Output :
[4, 5, 8, 10, 12, 13]

Answer:

```
function sumOfIndex(a,b){
   const sum=[]
   for(var i=0;i<a.length;i++){
      sum.push(a[i]+b[i])
   }
   return sum
}
console.log(sumOfIndex([1,2,3,4],[5,6,7,8]));
```

7. Write a JavaScript program to find a pair of elements (indices of the two numbers) in a given array whose sum equals a specific target number.
Input: numbers= [10,20,10,40,50,60,70], target=50 Output: 2, 3

Answer:

```
function pairs(a, target) {
   const indices = {};
   for ( var i=0; i < a.length; i++) {
      indices[a[i]] = i;
   }
   let low = 0;
   let upp = a.length - 1;
   while (low < upp) {
      var sum = a[low] + a[upp];
      if (target === sum) {
         console.log(`${indices[a[low]]}, ${indices[a[upp]]}`);
         low++;
         upp--;
      } else if (target > sum) {
         low++;
      } else {
         upp--;
      }
   }
}
pairs([10,20,10,40,50,60,70], 50);
```

8. What will be printed to the console and explanation (Closures problem)

```
for (var i = 0; i < 3; i++) {
  setTimeout(function() {
```

```
    console.log(i);
  }, 1000);
}
```

Answer:

Output: 3

3

3

The setTimeout function is called during each loop iteration, but because setTimeout is asynchronous, The setTimeout function waits for 1 second before running its callback. By the time, loop will finish its iterations and the value of i will be 3.

9. output for below and why?

```
console.log(0 == '0');
console.log(0 === '0');
console.log(null == undefined);
console.log(null === undefined);
```

Answer:

Output :

true

false

true

false

'==' only checks for value, in first and third cases , the values are same , so it will return a true statement.

But '===' checks for both value and type, since 0 is a number and '0' is a string , eventhough their values are same , their types are different, so it will return false. Same goes for 4th case, type of null is an object and type of undefined in undefined.

10. Event Loop and Microtasks

What will be printed and why?

```
console.log('Start');
setTimeout(() => {
  console.log('Timeout 1');
}, 0);
Promise.resolve().then(() => {
  console.log('Promise 1');
```

```
});

setTimeout(() => {

  console.log('Timeout 2');

}, 0);

Promise.resolve().then(() => {

  console.log('Promise 2');

});

console.log('End');
```

Answer:

Output:

Start

End

Promise 1

Promise 2

Timeout 1

Timeout 2

Since console.log('Start'); **and** console.log('End'); **are synchronous, it will get executed first. After it's execution, the event loop checks the microtask queue which has promises. After all the microtask execution,then macro task execution will happens which has setTimeout callback.**