Character Occurrences in a String:

```
//Count number of character occurences in a string

function charCount(str) {
  let obj = {};
  for (let i = 0; i < str.length; i++) {
    let char = str[i].toLowerCase();
    if (/[a-z0-9]/.test(char)) {
      if (obj[char] > 0) {
        obj[char]++;
      } else {
        obj[char] = 1;
      }
    }
  }
  return obj;
}


function charCount1(str) {
    let obj = {};
    for (let char of str) {
      char = char.toLowerCase();
      if (/[a-z0-9]/.test(char)) {
        if (obj[char] > 0) {
          obj[char]++;
        } else {
          obj[char] = 1;
        }
      }
    }
    return obj;
  }


  function charCount2(str) {
    let obj = {};
    for (let char of str) {
      char = char.toLowerCase();
      if (/[a-z0-9]/.test(char)) {
        obj[char] = ++obj[char] || 1;
        //It takes a char and accesses the corresponding value in our object
        //and if the value exists then we are going to add 1 (obj[char] = 1) or if it's 
false we are going to set it to one
      }
    }
    return obj;
  }



//The charCodeAt() method returns the Unicode of the character at a specified index 
(position) in a string.
```

```
//The charAt() method returns the character at a specified index (position) in a string.
const charCount3 = (str) => {
    let obj = {};
    for (let char of str) {

      if (isAlphaNumeric(char)) {
        char = char.toLowerCase();
        obj[char] = ++obj[char] || 1;

      }
    }
    return obj;
  }

const isAlphaNumeric = (char) => {
    let code = char.charCodeAt(0);
    if(
        !(code> 47 && code < 58) && //numeric (0-9)
        !(code > 64 && code < 91) && //upper alpha (A-Z)
        !(code > 96 && code < 123) //lower alpha (a-z)
    ){
        return false;
    }
    return true;
}


console.log('*********For Loop***********', charCount('Hello11 hi!'))

console.log('******For OF********', charCount1('Hello11 hi!'))

console.log('******For OF With || ********', charCount2('Hello11 hi!'))

console.log('******For OF With || and charCodeAt with ES6 ********', charCount3('Hello11
hi!'))
```

Sorting 0s, 1s, and 2s:

```
function Sort012 (arr,n) {
    let count0 = 0
    let count1 = 0
    let count2 = 0
    for(let i =0; i<n; i++) {
        if(arr[i] === 0) count0++;
        if(arr[i] === 1) count1++;
        if(arr[i] === 2) count2++;
    }
    let arr1 = []
    for(let i =0; i<count0; i++) {
        arr1.push(0)
    }
    for(let i =0; i<count1; i++) {
        arr1.push(1)
    }
    for(let i =0; i<count2; i++) {
        arr1.push(2)
    }
    return arr1
}

console.log(Sort012([0,1,0,2,2,1,1,0,0,0,0],5))
```

Negative Numbers to one side in an Array:

```
//Time Complexity : O(n)
//Space Complexity : O(n)
function moveNegative(arr){
    let len = arr.length
    let negative = []
    let postivie = []
    for(let i=0; i<len; i++){
        if(arr[i]<0){
            negative.push(arr[i])

        }else if(arr[i]>0){
            postivie.push(arr[i])
        }

    }
    return [...negative, ...postivie]
}
console.log("moveNegative========>",moveNegative([2,3,4,-1,-3-5,-2,31,45]))


//Time Complexity : O(n) Splice Takes: O(n^2)
//Space Complexity : O(n)
function moveNegativeSplice(arr) {
    let negative = [];
    for (let i = 0; i < arr.length; ) {
        if (arr[i] < 0) {
            negative.push(arr[i]);
            arr.splice(i, 1);
        } else {
            i++;  // Increment i only if no element is removed
        }
    }
    return [...negative, ...arr];
}

console.log("moveNegativeSplice========>",moveNegativeSplice([2, 3, 4, -1, -3, -5, -2, 31,
45]));


//Two-pointer technique
function moveNegativeOptimized(arr) {
    let left = 0, right = arr.length - 1;
    while (left <= right) {
        if (arr[left] < 0) {
            left++;
        } else if (arr[right] >= 0) {
            right--;
        } else {
            // Swap arr[left] and arr[right]
            [arr[left], arr[right]] = [arr[right], arr[left]];
            left++;
            right--;
        }
    }
    return arr;
}
console.log("moveNegativeOptimized========>",moveNegativeOptimized([2, 3, 4, -1, -3, -5, -
2, 31, 45]));
```