Javascript Problems

Flattern Array

Valid parentheses

Group Anagram

isPanagram

Auto retry on error

Flattern Object

Missing Number

Majority Number

Find duplicates in the string

Find the common elements in the string

Join comment element in a an array

Two sum

Intersection of two arrays

First unique char in string

Group by property

Rotate an array

isPalindrome

Flattern Array

```
function flatternArray(arr, depth = 0) {
  let newArray = []

  arr.forEach((item) => {
    if (Array.isArray(item) && depth > 0) {
      newArray.push(...flatternArray(item, depth-1))
    } else {
      newArray.push(item)
    }
  })
  return newArray
}
const array = [1, 2, [3, 4  ,[5]]]
console.log(flatternArray(array , 2))  //[1 , 2 , 3 ,4 ,5]
```

Valid Parentheses

```
function validParentheses(string) {
    let stack = [];
    for (let str of string) {
      switch (str) {
        case "(":
          stack.push(")");
          break;
        case "{":
          stack.push("}");
          break;
        case "[":
          stack.push("]");
          break;
        default:
          if (stack.pop() !== str) return false;
      }
    }
    return stack.length === 0;
  }

  console.log(validParentheses("()[]{}")); //true
```

Group Anagram

```
function groupAnagram(arr) {
  let newObj = {};

  for (let i = 0; i < arr.length; i++) {
    const sorted = arr[i].split("").sort().join("");
    if (!newObj[sorted]) {
      newObj[sorted] = [arr[i]];
    } else {
      newObj[sorted].push(arr[i]);
    }
  }
  return Object.values(newObj);
}

  console.log(groupAnagram(["eat", "tea", "tan", "ate", "nat", "bat"]));
//   [["eat", "tea", "ate"], ["tan", "nat"], ["bat"]]
```

iSPanagram

```
function isPanagram(sentence) {
    let map = new Map();
    for (let i = 0; i < sentence.length; i++) {
      if (!map.has(sentence[i])) {
        map.set(sentence[i], 1);
      }
      if (map.size === 26) {
        return true;
      }
    }
    return false;
  }


  // console.log(isPanagram("The quick brown fox jumps over the lazy lazy")); //true
```

Auto retry on error

```
function fetchWithRetry(fetcher, maxRetryCount) {
    return new Promise((resolve, reject) => {
      let retryCount = 0;
      const fetch = () => {
        fetcher()
          .then(resolve)
          .catch((error) => {
            if (retryCount <= maxRetryCount) {
              retryCount++;
              fetch();
            } else {
              return reject(error);
            }
          });
      };
      fetch();
    });
  }
```

InterSection Array

```
function interSection(nums1, nums2) {
    let duplicates = [];
    let set = new Set(nums1);
    for (let i = 0; i <= nums2.length; i++) {
      if (set.has(nums2[i])) {
        duplicates.push(nums2[i]);
      }
    }
    return duplicates;
  }


  console.log(interSection([1, 3, 2], [1, 2, 5])); //[1 ,2]
```

Join common Elements

```
function commonElements(arr) {
    const newObj = {};
    arr.forEach((element) => {
      if (newObj[element]) {
        newObj[element].push(element);
      } else {
        newObj[element] = [element];
      }
    });
    return Object.values(newObj);
  }


  console.log(commonElements([1, 2, 3, 2, 3, 4])); //[[1], [2, 2], [3, 3], [4]]
```

Join common elements String

```
function countElementsStr(string) {
    let newObj = {};
    string.split("").forEach((element) => {
      if (newObj[element]) {
        newObj[element]++;
      } else {
        newObj[element] = 1;
      }
    });
    return newObj;
  }
   console.log(countElementsStr("racecar"));  {
  a: 2,
  c: 2,
 e: 1,
  r: 2
}
```

Flattern Object

```
function flattenObj(arr) {
    let newObj = {};
    Object.keys(arr).forEach((element) => {
      if (typeof arr[element] == "object") {
        Object.assign(newObj, flattenObj(arr[element]));
      } else {
        newObj[element] = arr[element];
      }
    });
    return newObj;
  }

  const testObject2 = {
    name: "John",
    age: 25,
    address: {
      city: "New York",
      zip: "10001",
    },
    hobbies: ["reading", "coding"],
  };

   console.log(flattenObj(testObject2));
```

Get Element by Property

```
function getElmentByProperty(array, property) {
   const grouped = {};
   array.forEach((element) => {
     const value = element[property];
     if (!grouped[value]) {
       grouped[value] = [];
     }
     grouped[value].push(element.name);
   });
   return grouped;
 }
 const arr = [
   { name: "Alice", age: 25 },
   { name: "Bob", age: 30 },
   { name: "Charlie", age: 25 },
   { name: "David", age: 30 },
 ];
  console.log(getElmentByProperty(arr, "age"));
 /*  {
    25: ["Alice", "Charlie"],
    30: ["Bob", "David"]
  } */
```

TWO SUM

```
function TwoSum(nums1, nums2, target) {
    for (let i = 0; i <= nums1.length; i++) {
      for (let j = 0; j <= nums2.length; j++) {
        if (nums1[i] + nums2[j] === target) {
          return [nums1[i], nums2[j]];
        }
      }
    }
    return [];
  }

   console.log(TwoSum([1, 3, 4], [1, 2, 4], 6)); //[4, 2]
```

Find Duplicates in a string

```
function FindSuplicate(arr) {
  const result = [];
  arr.forEach((word) => {
    const obj = {};
    const duplicates = [];
    word.split("").forEach((letter) => {
      if (obj[letter]) {
        obj[letter]++;
      } else {
        obj[letter] = 1;
      }
      if (obj[letter] === 2) {
        duplicates.push(letter);
      }
    });
    result.push(duplicates.length > 0 ? duplicates.join("") : null);
  });
  return result;
}


 console.log(FindSuplicate(["hello", "world", "haii"])); //["l", null, "i"]
```

Find first unique letter in a string

```
function firstUnique(string) {
    const obj = {};
    string.split("").forEach((letter) => {
      if (!obj[letter]) {
        obj[letter] = 1;
      } else {
        obj[letter] += 1;
      }
    });
    for (let i = 0; i < string.length; i++) {
      if (obj[string[i]] === 1) {
        return string[i];
      }
    }
    return -1;
  }


   console.log(firstUnique("leetcode")); //"l"
```

Find the majority of elements in a array

```
function majorityElement(nums) {
    let element;
    let count = 0;

    nums.forEach((char) => {
      if (count === 0) {
        element = char;
      }
      if (char === element) {
        count += 1;
      } else {
        count -= 1;
      }
    });
    return element;
  }

   console.log(majorityElement([2, 2, 1, 1, 4, 3, 2])); //2
```

Rotate an array

```
function rotateArray(arr, target) {
    target %= arr.length;
    if (target === 0) return arr;
    const extractedArray = arr.splice(arr.length - target, target);
    arr.unshift(...extractedArray);
    return arr;
  }

   console.log(rotateArray([1, 2, 3, 4, 5], 3)); // [3, 4, 5, 1, 2]
```

Missing Number

```
function missignNumber(arr: any) {
  const n = arr.length + 1;
  const sum = (n * (n + 1)) / 2;
  const actualSum = arr.myReduce((sum: any, acc: any) => sum + acc, 0);
  return sum - actualSum;
}

// console.log(missignNumber([1, 3, 4, 5])); //2
```

Find the string is Palindrome or not

```
function isPalindrome(str) {
  let newStr = "";
  for (let i = str.length - 1; i >= 0; i--) {
    newStr += str[i];
  }
  return str === newStr;
}
// console.log(isPalindrome("racecar"));
```