<https://blog.csdn.net/qq_42946917/article/details/89024468>

**1.判断一段字符串是否为回文字符串**？

**回文字符串：即字符串从前往后读和从后往前读字符顺序是一致的。**

如：字符串abccba，从前往后读是a-b-c-c-b-a；从后往前读也是a-b-c-c-b-a

方法一: 1.字符串转数组;2.用reverse()函数颠倒;3.使用join(‘’)拼接成字符串

function checkPalindrom(str) {

return str == str.split('').reverse().join('');

}  
方法二: 从后面往前面遍历字符串，组装成新字符串与原字符串进行比较

function checkPalindrom (str){

var len = str.length; var str1 = "";

for(var i=len-1; i>=0;i--){

str1+=str[i];

}

return str1 == str

}

**2.数组去重**

方法一: 直接双重遍历：一层遍历至倒数第二项，二层遍历从一层的下一项起，重复的splice移除后--

var arr = [1,2,3,3,4,3,5,3,7,5];

function unique(arr) {

for(var i = 0; i < arr.length-1; i++){//一层遍历至倒数第二项

for(var j = i+1; j < arr.length; j++){//二次从一层的下一项开始

if(arr[i]==arr[j]){

arr.splice(j,1);//console.log(arr[j]);

j--; //减去移除项的影响

}

}

}

return arr;

}  
方法二: 用indexOf()方法：新建一个数组，遍历原数组，通过indexof判断，不存在于新数组中就push

function unique(arr) {

if (!Array.isArray(arr)) {

console.log('type error!')

return

}

var array = [];

for (var i = 0; i < arr.length; i++) {

if (array .indexOf(arr[i]) === -1) {

array .push(arr[i])

}

}

return array;

}方法三: 用filter方法：当前元素在原始数组中的第一个索引==当前索引值

function unique(arr) {

return arr.filter(function(item, index, self) {

//当前元素，在原始数组中的第一个索引==当前索引值，则返回当前元素

return self.indexOf(item, 0) === index;

});

}

方法四:使用ES6的Map和filter

function unique(arr) {

const res = new Map();

return arr.filter((a) => !res.has(a) && res.set(a, 1))

}

方法五:使用ES6的Set

function unique(arr) {

return Array.from(new Set(arr))

}

**3.对象数组去重（[{a:1},{b:1}]）**

方法1：利用对象访问属性的方法，判断对象中是否存在key

function unique (arr) {

var result = []; var obj = {};

for(var i =0; i<arr.length; i++){

if(!obj[arr[i].key]){

result.push(arr[i]);

obj[arr[i].key] = true;

}

}

return result

}

方法2：利用reduce方法遍历数组,reduce第一个参数是遍历需要执行的函数，第二个参数是item的初始值

function unique(arr){  
var obj = {};

result = arr.reduce(function(item, next) {

obj[next.key] ? '' : obj[next.key] = true && item.push(next);

return item;

}, []);

return result;

}

方法3：通过某个属性去重，传入两个值，prop为要依据去重的元素

function filterRepeat(arr,prop){

return arr.filter(function(element,index,self){

return self.findIndex(el=>el[prop]==element[prop])===index

})

}

**4.统计一个字符串出现最多的字母**

方法1：新建一个空对象，遍历循环字符串，通过charAt（i）获取元素，当对象中没有该元素时，该元素的值设为1，有时则值加一；然后再新建一个包含maxValue值和maxkey空数组的对象，遍历之前的对象，若遍历的值大于maxvalue则给maxvalue和maxkey中，如相等，则将k push到maxkey中

function getmaxObj(str) {

if (str.length == 1) {

return str;

}

var newObj = {};

for (var i = 0; i < str.length; i++) {

**if (!newObj[str.charAt(i)])**

**{ newObj[str.charAt(i)] = 1;}**

**else**

**{ newObj[str.charAt(i)] += 1;}**

**}**

**var maxObj ={**

**maxkey :[],** //若定义为字符串会出现是‘abc’时只输出a

**maxvalue:0**

**}**

**for (var k in newObj) {**

**if (newObj[k] > maxObj.maxvalue) {**

**maxObj.maxvalue = newObj[k];**

**maxObj.maxkey = [k];**

**}else if(newObj[k] === maxObj.maxvalue)**

**{ maxObj.maxkey.push(k);}**

**}**

**return maxObj;**

**}**

方法2：用变量存储目标字母和个数，循环遍历不断更新结果。知识点：

1、new Set(arr) //Set去重,目标字符串长的话去重还是有必要的

2.Array.filter() // 筛选得到符合条件的新数组

// 统计字符串中出现最多的字母与个数

const getStrMax = str => {

const arr = str.split("");//转成数组

const arrSet = new Set(arr);//去重

let maxCount = 0;//记录最大数量

let maxItem = [];//记录最大数量的字母，考虑到有重复，用了数组

for (let value of arrSet.values()) {

const count = arr.filter(item => {

return item === value

}).length;

if (count > maxCount) {

maxCount = count;

maxItem = [value];

} else if (count === maxCount) {

maxItem.push(value);

}

}

return {

maxItem,

maxCount

}

} **//方法三四输出的result[0]不能很好的识别‘abc’这种无重复的字符串**

方法3：先获取每个字母出现的个数，再取出个数最多的那个。知识点：

1、Array.map() //遍历原数组得到想要的新数组

2、[...new Set(arr)] //扩展运算符,用于set结构转数组

3、Array.sort() //数组排序

//获取每个字母的数量

const getResult = str => {

const arr = str.split("");//转成数组

return [...new Set(arr)].map(value => {

const count = arr.filter(item => {

return item === value

}).length

return {

value,

count

}

})

}

//按count从大到小排序

const result = getResult(testStr).sort((a, b) => {

return b.count - a.count

})

//第一个就是结果

console.log(result[0])

方法四：同方法三，只是应用reduce方法，减少了代码量。知识点：

1、Array.reduce() //遍历数组并累加指定的数据

2、Array.from() //类数组对象转数组

3、Object.entries() //对象转数组

const getByReduce = str => {

return Array.from(str).reduce((obj, v) => {

const objVal = obj[v] || 0;

objVal === 0 ? obj[v] = 1 : obj[v] = objVal + 1

return obj;

}, {});

}

//转数组然后排序

const arr = Object.entries(getByReduce(testStr)).sort((a, b) => {

return b[1] - a[1]

})

console.log(arr[0]);

**5.排序算法**

1、冒泡排序：

原理:每次比较相邻两数，小的交换到前面，每轮结束后最大的数交换到最后

var bubbleSort = function(arr) {

for (var i = 0, len = arr.length; i < len - 1; i++) {//遍历到倒数第二项

for (var j = i + 1; j < len; j++) { //从上一层的后面一个元素开始遍历

if (arr[i] > arr[j]) {

var temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

}

}

}

return arr;

};

或

function bubbleSort(arr) {

var len = arr.length;

for (var i = 0; i < len; i++) {

for (var j = 0; j < len - 1 - i; j++) {

if (arr[j] > arr[j+1]) { //相邻元素两两对比

var temp = arr[j+1]; //元素交换

arr[j+1] = arr[j];

arr[j] = temp;

}

}

}

return arr;

}

# [19道常见的JS面试算法题](https://www.cnblogs.com/djw12333/p/11647413.html)

<https://www.cnblogs.com/djw12333/p/11647413.html>