**389 find the difference**

Given two strings s and t which consist of only lowercase letters.

String t is generated by random shuffling string s and then add one more letter at a random position.

Find the letter that was added in t.

var findTheDifference = function(s, t) {

var q;

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

for(a=0;a<t.length;a++){

if(s[i]!=t[i]){

q=t[i];

}

else

{

s[i]=t[i]='';

}

}

if(q!==null){

console.log(q);

}

}

};

function findTheDifference(s, t) {

let c = t.charCodeAt(t.length-1);

for(let i=0; i < s.length; i++){

c = c ^ s.charCodeAt(i) ^ t.charCodeAt(i);

}

return String.fromCharCode(c);

};

charCodeAt() 方法可返回指定位置的字符的 Unicode 编码。

用^进行两个数组字符中的查找，正则表达式。

一开始用C++的想法实现，没有想到JS里自带的函数和其他方法的应用。

**412. Fizz Buzz**

Write a program that outputs the string representation of numbers from 1 to n.

But for multiples of three it should output “Fizz” instead of the number and for the multiples of five output “Buzz”. For numbers which are multiples of both three and five output “FizzBuzz”.

/\*\*

\* @param {number} n

\* @return {string[]}

\*/

var fizzBuzz = function(n) {

for(i = 1;i <= 15;i++){

var a=i;

if(i % 3 === 0 && i % 5 !== 0){

a="Fizz";

}

if(i % 3 !== 0 && i % 5 === 0){

a="Buzz";

}

if(i % 3 === 0 && i % 5 === 0){

a="FizzBuzz";

}

console.log（a）;

}

};

var fizzBuzz = function(n) {

var result = [],

str, i=1;

while( i <= n ){

str = "";

if( i%3===0 ) str = 'Fizz';

if( i%5===0 ) str += 'Buzz';

if(!str) str += i;

result.push( str );

i++;

}

return result;

}

if(!str) 用来判断str是否是空串，来代替if(str!=null && str!=undefined && str !='')

没有用三个if来判断而是用一种承接的关系通过增加字符串的字符来实现。

**1、Two Sum**

Given an array of integers, return indices of the two numbers such that they add up to a specific target.

You may assume that each input would have exactly one solution, and you may not use the same element twice.

**var twoSum = function(nums, target) {**

**var ret = [];**

**var exist = {};**

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

**if(typeof(exist[target - nums[i]]) !== 'undefined'){**

**ret.push(exist[target - nums[i]]);**

**ret.push(i + 1);**

**}**

**[nums[i]] = i + 1;**

**}**

**return ret**

**};**

Typeof()函数用来查看数据类型。

push() 方法可把它的参数顺序添加到 arrayObject 的尾部。它直接修改 arrayObject，而不是创建一个新的数组。

Exist[]判断对象是否存在

**151. Reverse Words in a String**

Given an input string, reverse the string word by word.

Given s = "the sky is blue",

return "blue is sky the".

var reverseWords = function(str) {

var a= new array();

a=split(s," ")

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

s[i]=a[a.length-i];

console.log(s[i])

}

}

前后倒置，通过split函数在空格处将每个单词分别分割出来，再通过倒置实现整句话的倒置。

**414. Third Maximum Number**

Given a non-empty array of integers, return the third maximum number in this array. If it does not exist, return the maximum number. The time complexity must be in O(n).

var largest = nums[0];

var third;

var second;

for(var i = 1; i < nums.length; i++){

if(nums[i] === largest || nums[i] === second || nums[i] === third) continue;

if(nums[i] > largest){

var temp1 = largest;

var temp2 = second;

largest = nums[i];

second = temp1;

third = temp2;

}else if(nums[i] > second || !second){

third = second;

second = nums[i];

} else if(nums[i] > third || !third){

third = nums[i];

}

}

if(!third) return largest;

return third;

通过循环进行多次比较。

心得：算法思路的简便性很重要，循环嵌套的应用很难和很广泛，对JS的基本语法仍然需要加大了解，一些常用的函数应该知道用法。基础不牢固导致写的很慢bug很多。数组的应用也很广泛，与其他语言有共通之处，其中重要的是思路其次是语法规范。