# 算法题：

* 两数之和

[1. 两数之和 - 力扣（LeetCode） (leetcode-cn.com)](https://leetcode-cn.com/problems/two-sum/submissions/)

|  |
| --- |
| function twoNum(nums , target){  var hash = new Map();  for(let i = 0 ;i < nums.length; i++){  if(hash.has(target-nums[i])){  return [hash.get(target-nums[i]),i];  }else {  hash.set(nums[i],i);  }  }  return []; } |

* 有效括号

[20. 有效的括号 - 力扣（LeetCode） (leetcode-cn.com)](https://leetcode-cn.com/problems/valid-parentheses/)

|  |
| --- |
| function valid(s){  if(s.length%2 !== 0){  return false;  }   const map = new Map([  [')','('],  [']','['],  ['}','{']  ]);  let stack = [];  for(let item of s){  if(!map.has(item)){ //item = ( { [  stack.push(item);  }else { //item = ) } ]  if(map.get(item) !== stack.pop()){  return false;  }  }  }   if(stack.length === 0){  return true;  } } |

* 数组去除/去重

[(4条消息) js 数组去除的几种方法\_活着就得有野心的博客-CSDN博客\_数组去除](https://blog.csdn.net/huoniu007/article/details/49529693)

|  |
| --- |
| var list = [0,1,1,2,4,2,6,8,3,1,0,6]  function unique( list )  {  let res = [];  for(let item of list){  if(res.indexOf(item) == -1){ //  res.push(item);  }  }  return res;  }  unique(list); |
| function deleteMore(inputList){  return inputList.filter((value,index,array)=>{  return array.indexOf(value) === index  }) } |
| 排序  var list = [0,1,1,2,4,2,6,8,3,1,0,6]  function unique4( list )  {  list.sort();  var re=[list[0]];  for(var i = 1; i < list.length; i++)  {  if( list[i] !== re[re.length-1])  {  re.push(list[i]);  }  }  return re;  }  unique4(list); |

* 12345678 转化成 RMB 形式：12,345,678

[19道常见的JS面试算法题 - tiana\_Z - 博客园 (cnblogs.com)](https://www.cnblogs.com/djw12333/p/11647413.html)

|  |
| --- |
| function RMB(str){  let arr = str.split('').reverse();  let res = [];  for(let i = 0; i < arr.length; i++){  res.push(arr[i]);  if((i+1)%3===0){  res.push(',');  }  }  return res.reverse().join("");  }  console.log(RMB("12345678999")) |

* 实现版本号比较

[js中版本号的比较 - 掘金 (juejin.cn)](https://juejin.cn/post/6844903942812336142)

|  |
| --- |
| var compareVersion = function (version1, version2) {      let v1 = version1.split("."), v2 = version2.split(".")      let n = v1.length, m = v2.length      let i = 0, j = 0      while (i < n || j < m) {          let a = 0, b = 0          if (i < n) a = parseInt(v1[i++])          if (j < m) b = parseInt(v2[j++])          if (a != b) return a > b ? 1 : -1      }      return 0  }; |

* 求最大公约数

[19道常见的JS面试算法题 - tiana\_Z - 博客园 (cnblogs.com)](https://www.cnblogs.com/djw12333/p/11647413.html)

|  |
| --- |
| function greatestCommonDivisor(a,b){  　　if (b === 0) {  　　return a;  　　} else {  　　return greatestCommonDivisor(b,a%b);  　　}  　　}; |

* 实现数组打乱

[如何将一个 JavaScript 数组打乱顺序 - 简书 (jianshu.com)](https://www.jianshu.com/p/4454eaf4bdf8)

|  |
| --- |
| Fisher–Yates shuffle 洗牌算法该方法就是每次在数组中随机产生一个位置，依次将数组中的每一项与该次产生的随机位置上的元素交换位置： var list = [19,5,0,6,9,78,50];  function noSequence(list){      let len = list.length;      let index, temp;      while(len>0){          index = Math.floor(Math.random()\*len);          temp = list[index];          list[index] = list[len-1];          list[len-1] = temp;          l--;      }      return list;  }  noSequence(list); |

* 斐波拉契

[js 实现斐波那契数列(数组缓存、动态规划、尾调用优化) - 简书 (jianshu.com)](https://www.jianshu.com/p/bbc7e54a98d6)

* 判断字符串回文

[19道常见的JS面试算法题 - tiana\_Z - 博客园 (cnblogs.com)](https://www.cnblogs.com/djw12333/p/11647413.html)

* sleep 函数

[(4条消息) javascript里的sleep()方法\_clschen的博客-CSDN博客\_js sleep()](https://blog.csdn.net/clschen/article/details/51727599)

* 查找数组中第二大元素

[(2条消息) 数组：找出数组中第二大的数\_加油呐的博客-CSDN博客\_求数组中第二大的数](https://blog.csdn.net/weixin_38108266/article/details/81269216)

|  |
| --- |
| var list = [19,5,0,6,9,78,50];  function findSec(list){      let max = list[0];      let sec = Number.MIN\_SAFE\_INTEGER;      for(let num of list){          if(num>max){              sec = max;              max = num;          }else if(num>sec){              sec = num;          }      }      return sec;  }  findSec(list); |

* 求数组的两个最大值
* 连续序列最长长度

[128. 最长连续序列 - 力扣（LeetCode） (leetcode-cn.com)](https://leetcode-cn.com/problems/longest-consecutive-sequence/submissions/)

set 去重+起点遍历（！set.has(item-1)）

|  |
| --- |
| ****输入：****nums = [100,4,200,1,3,2]  ****输出：****4  ****解释：****最长数字连续序列是 [1, 2, 3, 4]。它的长度为 4。  var longestConsecutive = function(nums) {      const set = new Set(nums);      let maxx = 0;      for(let item of nums){          if(!set.has(item-1)){              let cur = item;              let len = 1;              while(set.has(cur+1)){                  cur++;                  len++;              }              maxx = Math.max(maxx,len);          }      }      return maxx;  }; |

* 去掉字符串前后的空格

[JavaScript 正则表达式 | 菜鸟教程 (runoob.com)](https://www.runoob.com/js/js-regexp.html)

去除所有空格:

str = str.replace(/\s+/g,"")

去除两头空格:

str = str.replace(/^\s+|\s+$/g,"")

去除左空格：

str=str.replace( /^\s/, '')

去除右空格：

str=str.replace(/(\s$)/g, "")

* 最长前缀

[14. 最长公共前缀 - 力扣（LeetCode） (leetcode-cn.com)](https://leetcode-cn.com/problems/longest-common-prefix/submissions/)

|  |
| --- |
| var longestCommonPrefix = function(strs) {      if(strs.length==0) return "";      let res = strs[0];      for(let i = 1;i < strs.length;i++){          let j = 0;          for(;j<res.length && j<strs[i].length; j++){              if(res[j]!=strs[i][j]){                  break;              }          }          res = res.substring(0,j);      }      return res;  }; |

* 大数相乘（字符串相乘）

|  |
| --- |
| var multiply = function(num1, num2) {      let m = num1.length;      let n = num2.length;      let res = new Array(m+n).fill(0);      for(let i = m-1; i>=0; i--){          for(let j=n-1; j>=0; j--){              let mul = (num1[i]-0)\*(num2[j]-0);              let p1 = i+j;              let p2 = i+j+1;              let sum = res[p2]+mul;              res[p2] = sum % 10;              res[p1] += parseInt(sum/10);          }      }      let i = 0;      while(i<res.length&&res[i]==0){          i++;      }      let str = "";      for(;i<res.length;i++){          str += res[i];      }      return str.length == 0? "0":str;  }; |

* 朋友圈 (set)

|  |
| --- |
| obj={  a:[‘b’,’c’],  b:[‘a’],  c:[‘a’],  d:[]  }  输出2，有两个朋友圈（[a,b,c],[d]） |
| var obj = {  a:['b','c'],  b:['a','e'],  c:['a'],  d:[],  e:['b'],  f:[] }  function friendCircle(obj){  var circles = [];  for(let key in obj){  var bool = true;   for (let circle of circles){  if (circle.has(key)){  for(let item of obj[key]){ //[b,c]  circle.add(item);  }  bool = false;  }  }  if(bool){  var set = new Set(key);  for(let item of obj[key]){ //[b,c]  set.add(item);  }  circles.push(set);  }   }  console.log(circles);  console.log(circles.length); }  friendCircle(obj) |

* 判断素数

[5种你不知道的素数的判断方法 - 知乎 (zhihu.com)](https://zhuanlan.zhihu.com/p/104314640)

# 数据结构：

* 栈、队列 互相模拟

[如何用栈来模拟队列，如何用队列模拟栈？？\_x.yao的博客-CSDN博客\_用栈模拟队列](https://blog.csdn.net/lilililililiki/article/details/104439469)

* 二叉树前序遍历

[二叉树的后序遍历\_牛客题霸\_牛客网 (nowcoder.com)](https://www.nowcoder.com/practice/1291064f4d5d4bdeaefbf0dd47d78541?tpId=295&tags=&title=&difficulty=0&judgeStatus=0&rp=0&sourceUrl=/exam/oj?tab=%E7%AE%97%E6%B3%95%E7%AF%87&topicId=295)

[(2条消息) 详细介绍js实现二叉树非递归遍历算法(前序、中序、后序、层序、路径）\_coocochen的博客-CSDN博客](https://blog.csdn.net/qq_36368991/article/details/102652086)

|  |
| --- |
| 递归  function Preorder(root) {  let res = [];  const pre = (\_root) => {  if (\_root){  res.push(\_root.val)  pre(\_root.left)  pre(\_root.right)  }  }  pre(root)  return res } |
| 迭代  function PreOrder(root) {  let res = [];  if(root===null){  return [];  }  let tempStack = [root];  while (tempStack.length!==0){  //每次pop的都是一个左节点，然后下面再将右节点和左节点push进栈  //这样的话，每一次pop完之后，还没push进子节点时，stack栈里面都是右节点哦。  let node = tempStack.pop();  res.push(node.val);  //每次都把右节点先压进栈，之后把左节点叠上去  if(node.right){  tempStack.push(node.right)  }  if(node.left){  tempStack.push(node.left)  }  }  return res; } |

* 二叉树后序遍历

|  |
| --- |
| function postorder(root) {  let res = [];  const post = (\_root)=>{  if(\_root){  post(\_root.left)  post(\_root.right)  res.push(\_root.val)  }  }  return res; } |
| function postOrder(root) {  let res = [];  if (root === null){  return [];  }  let tempStack = [root];  while (tempStack.length > 0){  let node = tempStack.pop();  res.unshift(node.val); //unshift  if(node.left){  tempStack.push(node.left) //左右顺序跟前序相反  }  if(node.right){  tempStack.push(node.right)  }  }  return res; } |

* 二叉树中序遍历

|  |
| --- |
| function inorder(root) {  let res = [];  const ino = (\_root) => {  if (\_root){  ino(\_root.left);  res.push(\_root.val);  ino(\_root.right);  }  }  return res; } |
| function inOrder(root) {  let res = [];  if (root === null){  return [];  }  let tempStack = [];  while (root !== null || tempStack.length > 0){  while (root !== null){ //将根节点顺着左节点放进栈  tempStack.push(root);  root = root.left;  }    root = tempStack.pop();  res.push(root.val);   root = root.right; //换成右节点，之后再顺着这个右节点一直  }  return res; } |

* dfs 和bfs

[求二叉树的层序遍历\_牛客题霸\_牛客网 (nowcoder.com)](https://www.nowcoder.com/practice/04a5560e43e24e9db4595865dc9c63a3?tpId=295&tags=&title=&difficulty=0&judgeStatus=0&rp=0&sourceUrl=/exam/oj?tab=%E7%AE%97%E6%B3%95%E7%AF%87&topicId=295)

[二叉树的最大深度\_牛客题霸\_牛客网 (nowcoder.com)](https://www.nowcoder.com/practice/8a2b2bf6c19b4f23a9bdb9b233eefa73?tpId=295&tags=&title=&difficulty=0&judgeStatus=0&rp=0&sourceUrl=/exam/oj?tab=%E7%AE%97%E6%B3%95%E7%AF%87&topicId=295)

|  |
| --- |
| dfs  function maxDepth( root ) {  // write code here  const compute = (root) =>{  if(!root){return 0;}  let left = compute(root.left);  let right = compute(root.right);  return Math.max(left+1,right+1);  }    const dep = compute(root);  return dep;  } |
| bfs  function levelOrder( root ) {  // write code here  let res = [];  const pre = (root,level)=>{  if(root==null){return ;}  if(level>=res.length){  res.push([]);  }  res[level].push(root.val);  pre(root.left,level+1);  pre(root.right,level+1);  }  pre(root,0);  return res;  }  function levelOrderQue( root ){  let res = [];  let que = [root];  while(que.length!==0){  let tempRes = [];  let tempQue = [];  for(let i = 0; i < que.length; i++){  tempRes.push(que[i].val);  if(que[i].left){tempQue.push(que[i].left)}  if(que[i].right){tempQue.push(que[i].right)}  }  res.push(tempRes);  que = tempQue;  }  que = null;  return res;  } |

* 对称的二叉树

[剑指 Offer 28. 对称的二叉树 镜像，递归的思维非常简单！Javascript版 - 对称的二叉树 - 力扣（LeetCode） (leetcode-cn.com)](https://leetcode-cn.com/problems/dui-cheng-de-er-cha-shu-lcof/solution/jing-xiang-di-gui-de-si-wei-fei-chang-ji-gsrf/)

|  |
| --- |
| function mirrorBST(root) {  if (root === null){  return true;  }   const judge = (left,right) => {  if (!left && !right){  return true;  }  if (!left || !right){  return false;  }  return left.val === right.val && judge(left.left,right.right) && judge(left.right,right.left);  }   return judge(root.left,root.right); } |

* 快排

[面试官：说说你对快速排序的理解？如何实现？应用场景？ · Issue #275 · febobo/web-interview (github.com)](https://github.com/febobo/web-interview/issues/275)

|  |
| --- |
| var list = [0,1,15,9,4,2,26,8,3,12,20,6]  function quick( list )  {  const res = (arr) =>{  if(arr.length < 1){  return arr;  }  const left = [];  const right = [];  const mid = arr[0];  for (let i = 1; i < arr.length; i++){  if(arr[i] < mid){  left.push(arr[i]);  }  else{  right.push(arr[i]);  }  }  return [...res(left),mid,...res(right)];  }  return res(list);  }  quick(list); |

* 链表反转

[反转链表\_牛客题霸\_牛客网 (nowcoder.com)](https://www.nowcoder.com/practice/75e878df47f24fdc9dc3e400ec6058ca?tpId=295&tqId=23286&ru=/exam/oj&qru=/ta/format-top101/question-ranking&sourceUrl=/exam/oj?tab=%E7%AE%97%E6%B3%95%E7%AF%87&topicId=295)

# JS函数实现：

* 数组平铺/扁平flat

[js数组拍平(数组扁平化)的八种方式 - 掘金 (juejin.cn)](https://juejin.cn/post/6844903998223286285)

|  |
| --- |
| var arr = [1,2,[3,4,5,[6,7,8],9],10,[11,12]];  var arr1 = arr.toString().split(',').map((val)=>{  return parseInt(val)  });  console.log(arr1);  //[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]  //思路  [1,2,[3,4,5,[6,7,8],9],10,[11,12]].toString().split(",")  // ["1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11", "12"]  [1,2,[3,4,5,[6,7,8],9],10,[11,12]].toString()  //"1,2,3,4,5,6,7,8,9,10,11,12" |
| //选择扁平层次 function selectFlat(arr,level = 1) {  if(! arr instanceof Array){  return false;  }   let res = arr.reduce((pre,cur)=>{  if(Array.isArray(cur) && level>0){  return [...pre,...selectFlat(cur,level-1)]  }else {  return [...pre,cur]  }  },[])   return res; } console.log(selectFlat(arr,2)) |

* instanceof

|  |
| --- |
| function instanceOf(leftObj,rightFunction){  let leftProto = Object.getPrototypeOf(leftObj);  let rightPrototype = rightFunction.prototype;  while(true){  if(leftProto === rightPrototype){  return true  }else if(leftProto === null){  return false  }  leftProto = Object.getPrototypeOf(leftProto)  } } const array = [1,2,3,4,5,6] console.log(instanceOf(array,Array)) console.log(instanceOf(array,Object)) console.log(instanceOf(array,Number)) |

* new函数

|  |
| --- |
| function Person(name, age) {  this.name = name;  this.age = age; }  function newFunc(Func,...args) {  //new obj  let obj = {};  //obj.\_\_proto\_\_ = Func.prototype  Object.setPrototypeOf(obj,Func.prototype)  //this  let res = Func.apply(obj,args);  //constructor  return res instanceof Func ? res : obj; }  var ob = newFunc(Person,'name',10086); console.log(ob.name) console.log(ob.age) |

* bind函数
* url 解析 query

[获取地址中的query参数，完美版 - 简书 (jianshu.com)](https://www.jianshu.com/p/0f8167363148)

|  |
| --- |
| const urlString = 'https://www.example.com/test.html?a=param1&b=param2';//decodeURL(urlString)  function parseSearchParams(urlString){  const list = urlString.split('?');    if(list.length>1){  var obj = {};  var jquryList = list[1].split('&');  for(let i = 0; i < jquryList.length; i++){  let cur = jquryList[i].split('=');  obj[cur[0]] = cur[1];//decodeURLComponent(cur[1])  }  return obj;  }  return ;  }  parseSearchParams(urlString); |

* JS 闭包修改值

[JavaScript之更改闭包内的变量值 - Lowki - 博客园 (cnblogs.com)](https://www.cnblogs.com/lowki/p/10406320.html)

|  |  |
| --- | --- |
| |  | | --- | | 正确写法：  var foo = function(){      let val = 1;      function fo(){          console.log(val);          val++;      }      fo.reset = function (){          val = 1;      }      return fo;  }  var a = foo();  a();  a();  a.reset();  a(); | |

* 防抖节流

[JavaScript---节流与防抖 - 掘金 (juejin.cn)](https://juejin.cn/post/7030787304696315918)

[面试官：什么是防抖和节流？有什么区别？如何实现？ · Issue #83 · febobo/web-interview (github.com)](https://github.com/febobo/web-interview/issues/83)（防抖debounce、节流throttle）

|  |
| --- |
| **节流**  使用时间戳写法，事件会立即执行，停止触发后没有办法再次执行  function throttled1(fn, delay = 500) {  let oldtime = Date.now()  return function (...args) {  let newtime = Date.now()  if (newtime - oldtime >= delay) {  fn.apply(null, args)  oldtime = Date.now()  }  }  }  **使用定时器写法，delay毫秒后第一次执行，第二次事件停止触发后依然会再一次执行**  **function throttled2(fn, delay = 500) {**  **let timer = null**  **return function (...args) {**  **if (!timer) {**  **timer = setTimeout(() => {**  **fn.apply(this, args)**  **timer = null**  **}, delay);**  **}**  **}**  **}**  可以将时间戳写法的特性与定时器写法的特性相结合，实现一个更加精确的节流。实现如下  function throttled(fn, delay) {  let timer = null  let starttime = Date.now()  return function () {  let curTime = Date.now() // 当前时间  let remaining = delay - (curTime - starttime) // 从上一次到现在，还剩下多少多余时间  let context = this  let args = arguments  clearTimeout(timer)  if (remaining <= 0) {  fn.apply(context, args)  starttime = Date.now()  } else {  timer = setTimeout(fn, remaining);  }  }  } |
| **防抖**  简单版本的实现  **function debounce(func, wait) {**  **let timeout;**  **return function () {**  **let context = this; // 保存this指向**  **let args = arguments; // 拿到event对象**  **clearTimeout(timeout)**  **timeout = setTimeout(function(){**  **func.apply(context, args)**  **}, wait);**  **}**  **}**  防抖如果需要立即执行，可加入第三个参数用于判断，实现如下：  function debounce(func, wait, immediate) {  let timeout;  return function () {  let context = this;  let args = arguments;  if (timeout) clearTimeout(timeout); // timeout 不为null  if (immediate) {  let callNow = !timeout; // 第一次会立即执行，以后只有事件执行后才会再次触发  timeout = setTimeout(function () {  timeout = null;  }, wait)  if (callNow) {  func.apply(context, args)  }  }  else {  timeout = setTimeout(function () {  func.apply(context, args)  }, wait);  }  }  } |