1、分别简述队列、栈、堆的区别?

队列是先进先出：就像一条路，有一个入口和一个出口，先进去的就可以先出去。而栈就像一个箱子，后放的在上边，所以后进先出。堆是在程序运行时，而不是在程序编译时，申请某个大小的内存空间。即动态分配内存，对其访问和对一般内存的访问没有区别。

栈(Stack)是操作系统在建立某个进程时或者线程为这个线程建立的存储区域。在编程中，例如C/C++中，所有的局部变量都是从栈中分配内存空间，实际上也不是什么分配，只是从栈顶向上用就行，在退出函数的时候，只是修改栈指针就可以把栈中的内容销毁，所以速度最快。

堆(Heap)是应用程序在运行的时候请求操作系统分配给自己内存，一般是申请/给予的过程。由于从操作系统管理的内存分配所以在分配和销毁时都要占用时间，所以用堆的效率低的多!但是堆的好处是可以做的很大，C/C++对分配的Heap是不初始化的。

2、用JavaScript实现二分法查找

二分法查找，也称折半查找，是一种在有序数组中查找特定元素的搜索算法。查找过程可以分为以下步骤：

(1)首先，从有序数组的中间的元素开始搜索，如果该元素正好是目标元素(即要查找的元素)，则搜索过程结束，否则进行下一步。

(2)如果目标元素大于或者小于中间元素，则在数组大于或小于中间元素的那一半区域查找，然后重复第一步的操作。 (3)如果某一步数组为空，则表示找不到目标元素。

```
//二分搜索
//A为已按"升序排列"的数组，x为要查询的元素
//返回目标元素的下标
function binarySearch(A, x) {
  var low = 0, high = A.length - 1;
  while (low <= high) {
    var mid = Math.floor((low + high) / 2); //下取整
    if (x == A[mid]) {
      return mid;
    }
    if (x < A[mid]) {
      high = mid - 1;
    }
    else {
      low = mid + 1;
    }
  }
  return -1;
}
```

3、用JavaScript实现数组快速排序

关于快排算法的详细说明，可以参考阮一峰老师的文章快速排序 “快速排序”的思想很简单，整个排序过程只需要三步：

(1)在数据集之中，选择一个元素作为”基准”(pivot)。

(2)所有小于”基准”的元素，都移到”基准”的左边;所有大于”基准”的元素，都移到”基准”的右边。

(3)对”基准”左边和右边的两个子集，不断重复第一步和第二步，直到所有子集只剩下一个元素为止。

方法一(尽可能不用js数组方法）：

```
function quickSort(arr){
 qSort(arr,0,arr.length - 1);
}
function qSort(arr,low,high){
 if(low < high){
  var partKey = partition(arr,low,high);
  qSort(arr,low, partKey - 1);
  qSort(arr,partKey + 1,high);
 }
}
function partition(arr,low,high){
 var key = arr[low]; //使用第一个元素作为分类依据
 while(low < high){
  while(low < high && arr[high] >= arr[key])
   high--;
  arr[low] = arr[high];
  while(low < high && arr[low] <= arr[key])
   low++;
  arr[high] = arr[low];
 }
 arr[low] = key;
 return low;
}
```

方法二（使用js数组方法）：

```
function quickSort(arr){
 if(arr.length <= 1) return arr;
 var index = Math.floor(arr.length/2);
 var key = arr.splice(index,1)[0];
 var left = [],right = [];
 arr.forEach(function(v){
  v <= key ? left.push(v) : right.push(v);
 });
 return quickSort(left).concat([key],quickSort(right));
}
```

方法三：递归法

```
function quickSort(arr){
            if(arr.length<=1){
                return arr;//如果数组只有一个数，就直接返回；
            }

            var num = Math.floor(arr.length/2);//找到中间数的索引值，如果是浮点数，则向下取整

            var numValue = arr.splice(num,1);//找到中间数的值
            var left = [];
            var right = [];

            for(var i=0;i<arr.length;i++){
                if(arr[i]<numValue){
                    left.push(arr[i]);//基准点的左边的数传到左边数组
                }
                else{
                   right.push(arr[i]);//基准点的右边的数传到右边数组
                }
            }

            return quickSort(left).concat([numValue],quickSort(right));//递归不断重复比较
        }

        alert(quickSort([32,45,37,16,2,87]));//弹出“2,16,32,37,45,87”
```

4、编写一个方法 求一个字符串的字节长度

假设：一个英文字符占用一个字节，一个中文字符占用两个字节

```
function GetBytes(str){

```

```
        var len = str.length;

        var bytes = len;

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

            if (str.charCodeAt(i) > 255) bytes++;

        }

        return bytes;

    }

alert(GetBytes("你好,as"));
```

5、找出下列正数组的最大差值

> 输入 [10,5,11,7,8,9]
>
> 输出 6

```
function getMaxProfit(arr) {

    var minPrice = arr[0];
    var maxProfit = 0;

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

        minPrice = Math.min(minPrice, currentPrice);

        var potentialProfit = currentPrice - minPrice;

        maxProfit = Math.max(maxProfit, potentialProfit);
    }

    return maxProfit;
}
```

6、判断一个单词是否是回文?

> 什么是回文?

> 回文是指把相同的词汇或句子，在下文中调换位置或颠倒过来，产生首尾回环的情趣，叫做回文，也叫回环。比如 mamam redivider .

```
function checkPalindrom(str) {
    return str == str.split('').reverse().join('');
}
```

```
// while loop
const isPalindromicB = (w) => {

    let len = w.length;
    // 感谢 @拉比克魔王 的指点
    let start = Math.ceil(len / 2);
    while (start < len) {
      if (w[start] !== w[len - start - 1]) {
            return false;
       }
       start++;
    }
    return true;
};
```

7、如何消除一个数组里面重复的元素?

基本数组去重

```
Array.prototype.unique = function(){
 var result = [];
 this.forEach(function(v){
  if(result.indexOf(v) < 0){
   result.push(v);
  }
 });
 return result;
}
```

利用hash表去重，这是一种空间换时间的方法

```
Array.prototype.unique = function(){
 var result = [],hash = {};
 this.forEach(function(v){
  if(!hash[v]){
   hash[v] = true;
   result.push(v);
  }
 });
 return result;
}
```

上面的方法存在一个bug，对于数组[1,2,'1','2',3]，去重结果为[1,2,3]，原因在于对象对属性索引时会进行强制类型转换，arr['1']和arr[1]得到的都是arr[1]的值，因此需做一些改变：

```
Array.prototype.unique = function(){
 var result = [],hash = {};
 this.forEach(function(v){
  var type = typeof(v); //获取元素类型
  hash[v] || (hash[v] = new Array());
  if(hash[v].indexOf(type) < 0){
   hash[v].push(type); //存储类型
   result.push(v);
  }
 });
 return result;
}
```

先排序后去重

```
Array.prototype.unique = function(){
 var result = [this[0]];
 this.sort();
 this.forEach(function(v){
  v != result[result.length - 1] && result.push(v); //仅与result最后一个元素比较
 });
}
```

8、统计字符串中字母个数或统计最多字母数

输入： afjghdfraaaasdenas

输出： a

```
function findMaxDuplicateChar(str) {
    if(str.length == 1) {
        return str;
```

```
    }
    let charObj = {};
    for(let i=0;i<str.length;i++) {
        if(!charObj[str.charAt(i)]) {
            charObj[str.charAt(i)] = 1;
        }else{
            charObj[str.charAt(i)] += 1;
        }
    }
    let maxChar = '',
    maxValue = 1;
    for(var k in charObj) {
        if(charObj[k] >= maxValue) {
            maxChar = k;
            maxValue = charObj[k];
        }
    }
    return maxChar;
}
```

9、随机生成指定长度的字符串

```
function randomString(n) {
  let str = 'abcdefghijklmnopqrstuvwxyz9876543210';
  let tmp = '',
      i = 0,
      l = str.length;
  for (i = 0; i < n; i++) {
    tmp += str.charAt(Math.floor(Math.random() * l));
  }
  return tmp;
}
```

10、写一个isPrime()函数，当其为质数时返回true，否则返回false。

首先， 因为JavaScript不同于C或者Java，因此你不能信任传递来的数据类型。如果面试官没有明确地告诉你，你应该询问他是否需要做输入检查，还是不进行检查直接写函数。严格上说，应该对函数的输入进行检查。

第二点要记住：负数不是质数。同样的，1和0也不是，因此，首先测试这些数字。此外，2是质数中唯一的偶数。没有必要用一个循环来验证4,6,8。再则，如果一个数字不能被2整除，那么它不能被4，6，8等整除。因此，你的循环必须跳过这些数字。如果你测试输入偶数，你的算法将慢2倍（你测试双倍数字）。可以采取其他一些更明智的优化手段，我这里采用的是适用于大多数情况的。例如，如果一个数字不能被5整除，它也不会被5的倍数整除。所以，没有必要检测10,15,20等等。

最后一点，你不需要检查比输入数字的开方还要大的数字。我感觉人们会遗漏掉这一点，并且也不会因为此而获得消极的反馈。但是，展示出这一方面的知识会给你额外加分。

现在你具备了这个问题的背景知识，下面是总结以上所有考虑的解决方案：

```
function isPrime(number) {
 // If your browser doesn't support the method Number.isInteger of ECMAScript 6,
 // you can implement your own pretty easily
 if (typeof number !== 'number' || !Number.isInteger(number)) {
  // Alternatively you can throw an error.
  return false;
 }
 if (number < 2) {
  return false;
 }
 if (number === 2) {
  return true;
 } else if (number % 2 === 0) {
  return false;
 }
 var squareRoot = Math.sqrt(number);
 for(var i = 3; i <= squareRoot; i += 2) {
  if (number % i === 0) {
   return false;
  }
 }
 return true;
}
```

11、有100格台阶，可以跨1步可以跨2步，那么一个有多少种走法

本质是斐波那契算法

```
var Stairs = new step();
function step(){
  this.n1=1;
  this.n2=2;
```

```
  this.total=100;
  this.getFunction = getFunction;
}
function getFunction(){
    for(i=2;i<this.total;i++){
      res = this.n1 + this.n2;
      this.n1 = this.n2;
      this.n2 = res;
    }
  return res;
}
var totalStairs = Stairs.getFunction();
alert(totalStairs)
```