一，js的三种书写位置  
1：行间：js写在元素的身上，作为元素的属性  
     如：<input type="button" value="按钮" onclick="alert('我被点击了')" />  
写在行间：  
     优点：  
        直接  
     缺点：  
        不易阅读  
        不能复用，修改麻烦，维护麻烦  
2：内嵌：js代码写在一个固定的<script></script>标签对中  
   内嵌：  
     优点：可以复用，容易维护，方便修改  
   缺点：  
     不能跨文件使用。  
书写方式：<input type="button" value="按钮" id="btn" />  
          <script>  
               document.getElementById("btn").onclick = function (){  
                    alert("我被点击了")      
               };  
          </script>  
3:外链：用script的src进行引入的src="文件路径"  
     如：<script src="demo.js"></script>  
4：如果要把script放在head中，js代码有获取元素的，需要加上window.onload=function(){}  
  
二，变量  
1，变量：可变的量 作用：存储数据  好处：可复用数据，操作数据简洁  
声明变量：关键字var 格式：var 变量名  
       赋值：变量=值（把右边的值赋给左边的变量）  
如：var btn=document.getElementById("btn");  
2，变量的命名规则（变量名要有语义化）  
允许使用数字，字母，下划线 美元符组合而成  
     不能以数字开头  
     不允许使用关键字和保留字  
     关键字：当前语法中正在使用的单词  
          var function window for if                     
          http://www.w3school.com.cn/js/pro\_js\_keywords.asp  
               保留字：将来可能在语法中使用的单词  
                    http://www.w3school.com.cn/js/pro\_js\_reservedwords.asp  
                    class  
3，变量的命名风格  
     命名风格：  
          小驼峰  
             当几个单词组合命名，从第二个单词开始，首字母大写  
          大驼峰：  
             当几个单词组合命名，所有单词首字母大写  
               小驼峰var firstElementBtn;  
               大驼峰var FirstElementBtn  
4.声明变量的时候可以不用var,当不使用var声明变量的时候，直接进行赋值，程序会自动创建这个变量.  
     注意：a.在函数内容不使用var声明变量，会成为全局的变量  
           b.声明变量的时候，一定带上var  
  
  
三，函数  
函数：可放多条语句  作用：复用代码  
声明函数：关键字 function 完整语法：  
          function 函数名字（）{  
               代码块  
          }  
函数名字遵循变量的命名规则，只有函数被调用了代码才会被执行  
函数调用：直接调用，调用的是有名函数。  
     事件调用，调用的是匿名函数。  
  
  
  
四，属性的操作  
两种操作：赋值操作：元素.属性名=值 如：box.style.width=100px;  
     取值操作：元素.属性名  如：alert（box.style.width）;  
属性操作的方式：  
     1用（.）的方式  如：box.style.width  
     2用[]的方式     如：box["style"]["font-size"]  
属性操作的样式的复合样式写法：box.style.cssText = "width:500px;height:500px;background:yellow;";（cssText会覆盖之前行间中的样式）  
属性名的命名规则：1遇到复合样式：font-size，background-color时，使用（.）方式操作时，把"-"去掉，"-"后面的字母大写。  
               如：box.style.fontSize  
          2,一些属性名和系统规定有冲突，那么是用另外一个词来代替这个属性名。  
               如：class可用className代替  
注意：属性名的命名规则与变量名的命名规则一样。  
获取属性值：id class->className value(input texttrea select等) href/src innerHTML  
     href/src获取的是绝对路径  
     innerHTML获取的是某个元素中的全部结构（内容）。  
注意：1元素身上的style属性，使用style操作样式的时候，赋值操作是写在行间的，取值操作也是取行间样式的值。  
      2属性可读可写。  
      3使用class时，class要改为className来代替；  
      4重新赋值，会把之前的class覆盖掉  
五，字符串  
     用一对单引号或者双引号包起来的0个或多个字符组成的串（单引号或双引号必须是成对出现）。  
        字符串长度：字符串.lengt 如：str4.length。  
     字符串拼接（连接）：把两个字符串连接成一个字符串。  
     连接符是：+号  
     注意：当加号的两边有一个为字符串，那么加号的作用就是拼接字符串  
           两边都是数字，那么做加法运算  
  
六  
1.if流程控制语句：当有不同的情况，执行不同的代码，达到不同的效果  
     语法：1：if(条件){  
               当条件成立的时候，执行这里面的代码  
          }  
           2：  
          if(条件){  
               当条件成立的时候，执行这里面的代码  
          }else{  
               当条件不成立的时候，执行这里的代码  
          }  
     注意：1：if根据条件成立与否执行不同的代码  
              2：在条件执行后获得到一个值成为布尔值；有两个：为true时则为真，为false时则为假  
              3：一个=号是赋值的意思，==为判断是否等于，判断两边是否相同，相同则为true，不同则为false。  
              4：程序首先会把if后括号中的判断条件转为true或者false，然后交给if去做判断  
2.for流程控制语句：根据一定的条件，重复的执行一行或多行代码  
     语法：  
          for (条件初始化; 条件判断; 条件的变化) {  
             当条件判断满足真的时候，重复执行的代码  
          }  
     执行流程：  
          第一步：条件初始化  
          第二步：条件判断  
          第三步：当条件判断满足真的时候，重复执行的代码  
          第四步：执行条件变化  
          重复 第二步、第三步、第四步，直到条件判断为假的时候，整个for循环就执行结束了  
  
  
七，数组： (一堆)数据的集合  
  
        var 变量名 = [];  
         
        数组中存放了多个数据，每个数据以逗号隔开，最后一个数据后面不加逗号  
        如果数组中没有数据，称之为空数组  
  
        数组中数据按照一定的规则进行排列，系统会为每一个数据打上一个编号，默认编号是从"0"开始的  
  
        编号会作为数组的属性，通过属性就可以取到对应的值  
  
        编号 也叫作下标  
  
        编号都是字符串，当这个编号为纯数字，可以直接写成数字  
  
        取数组中第一项：arr[0]  
        取数组的最后一项：arr[arr.length-1]  
  
        获取数组数据的个数，数组的长度：数组.length  
  
  
八.获取元素的方法  
1.通过id获取元素  
     document.getElementById("id名字")；注意：通过ID获得元素，必须是document  
2.通过标签名获取元素  
     document.getElementsBytTagName("标签名");  
     注意：1.通过标签名获取的是这一类元素的一个集合，如果要获得某一个具体的元素时，要通过下标的方式来操作。如：取第一个li allLi[0]  
           2.从某个元素下获取某一类元素时写法：元素.getElementsByTagName("标签名")  
3.通过class名字获取元素  
     document.getElementsByClassName("class名字")  
     注意：1.通过classNames获取的元素存放在集合中  
           2.获取的这个集合只有一项时，获取里面的元素也要用下标的方式  
4.通过css的选择器  
     通过CSS 选择器获取一组元素  
          document.querySelectorAll("css选择器");  
          元素.querySelectorAll("css选择器");  
     通过CSS 选择器获取一个元素 获取的是第一个元素  
          document.querySelector("css选择器")  
          元素.querySelector("css选择器")            
  
九.关键字：  
     1.this  
          谁调用了函数，this的值就指向谁！！！  
               1. 直接调用this指向window  
  
               2. 事件调用指向触发这个事件的元素  
  
十.数据类型：区分不同的数据类型，当操作数据的时候，根据不同类型的数据会得到不同的结果。  
     1.js中的数据类型：string（字符串）number（数字）Boolean(布尔值) Undefined（未定义） null（空） object(复杂数据类型)  
     a.string:由一对单引号或双引号包含的零个或多个字符 组成的串  
     b.number:0-9的数字组成（Infinity最大的数字表示）  
          \*\*parseInt(字符串)  
               注意：从左向后查找，首字符不是数字，那么返回NaN，  
                        如果首字符十个数字，继续向后查找知道碰到不是数字的字符，返回找到的数字。  
          \*\*parsefloat（字符串）  
               注意：转换同parseInt  
                     同时返回小数  
     c.Boolean:true/false  
     d.Undefined:声明一个变量，并未赋值，默认值为undefined/对象身上没有这个属性，为undefined  
     e.object:复合类型是由简单和复合的数据类型组成的  
          注意：格式 属性:值  可以写多个，由逗号分开，最后一没有逗号  
                对象里的key值可以是数字。如：2：abc  
               属性可以加引号，也可以不用加引号，不符合命名规则的时候必须加引号  
               obj.height="100px";  
     \*\*\*1.isNaN(数据)  
          判断一个值是不是NaN  
          内部会调用Number方法，把数据转成数字  
  
         2.0 空字符串  null undefined NaN  转成false  
  
十一.typeof:检测数据的数据类型，返回这种类型的字符串“表示”  
     数据类型用typeof检测对应字符串  
          数据类型        typeof  
          String          "string"   
          Number          "number"  
          Boolean         "boolean"  
          Null            "object"  
          Undefined       "undefined"  
          Object          "object"  
  
          null用typeof返回的是object  
          用typeof 函数 返回的是"function"  
  
十二.运算符：  
1.算数运算符：+ 加、- 减、\* 乘、/ 除、% 取模（求余数）、++、--  
     注意：++/-- 前置  先自增（自减） 再赋值  
                 后置  先赋值 再自增（自减）  
  
2.赋值运算符：=、+=、-=、\*=、/=、%=  
     注意：+=是先运算后赋值 如：var a=1;a+=10元算结果是a=11;a+=10相当于a=a+10;  
  
3.关系运算符：<、>、<=、>=、==、!=、===、!==（运算比较后返回的都是布尔值）  
     注意：== 比较两边的值是否相同  
           === 先比较数据类型，不相同返回false；如果数据类型相同，再比较值是否相同。  
  
4.逻辑运算符：&& 与、|| 或、! 否 、三目运算符  
     注意：&& 并且   如果左边为真，返回右边的值；如果左边为假，返回左边的值。  
           || 或  如果左边为真，返回左边的值；如果左边为假，返回的是右边的值。  
            ！否 取反 非  
          三目运算符：书写格式（? : ）条件为真 返回？后面的值 否则返回：后面的值  
          注解：如果条件为真返回问号后面的值，否则返回的冒号后面的值。  
  
十三.传参  
   定义函数  
     function 函数名([参数]){  
          //写要重复执行的代码  
     }  
  
     []在写函数的参数的时候，表示的是可选的参数  
  
     函数定义的方式：  
          函数声明  
          function 函数名(){  
           
          }\*\*\*\*在当前的作用域中任何一个地方都可以调用  
          在当前的作用域中任何一个地方都都可以调用  
  
          函数表达式  
               var 变量 = function(){  
                                    
               }\*\*\*\*\*\*必须在赋值完成后才能调用，可以把声明的变量和函数声明，提升到当前作用域的顶端。  
     函数在一个对象中，也叫这个对象的方法。  
          对象.属性名（）;  
参数：分为两种：形参和实参  
     1.形参：形式上的参数。  
          定义函数的时候，写在()中  
          多个形参用逗号隔开，最后一个不用逗号  
          定义的形参就是这个函数内部的变量，只能在当前函数中使用  
          形参的值是在函数调用的时候，传入的值,没有接收到值，默认的值为undefined  
     2.实参：实际传入的值  
          函数调用写入在()中的值  
          值有多个用逗号隔开，最后一个值后面不用逗号  
          实参和形参是一一对应的  
使用参数需注意：  
     a.在函数的执行过程中，传入一些数据，使函数的灵活性和扩展性更大  
     b.想要在函数中使用数据的时候，要在定义函数的时候，写上要接收数据的形参  
     c.在调用函数的时候，形参和实参是一一对应的，如果在调用函数没有传入对应的实参，形参的值为undefined  
  
arguments:对应的值是一个类数组集合。（实参的集合）  
     注意：a.当实参不确定的时候用arguments。  
           b.只能在函数中使用  
           c.类数组，保存实际传入的参数的集合  
           d.拥有length属性  
           e.可用下标获取实际参数中对应的值  
  
return返回值：函数执行完成之后，返回的数据结果  
     注意：a.每一个函数调用后都有返回值，不使用return 函数的返回值为undefined  
           b.只能在函数中使用，return后面跟的值为函数执行后的返回值  
           c.如果只写一个return后面没有值，返回值为undefined  
           d.return可以停止函数继续执行  
  
           
十四.作用域：变量或函数可访问的范围（js中是用函数来划分作用域的）。  
     分别为全局作用域和局部作用域  
     1.全局作用域：一个变量或一个函数在非函数中声明，全局都可以访问。  
     2.局部作用域：一个变量或者函数在一个函数内声明  
     注意：a.函数中是可以声明另一个函数的，函数可以嵌套函数  
           b.在函数中声明的变量或者函数，只能在当前这个函数中访问  
           c.一个变量或者函数在另一个函数声明的时候，在当前这个函数中可以访问，包括这个函数中的子函数  
          如：function fn(){  
               var a = 10;  
               function fn2(){  //嵌套的函数，可以访问到声明这个函数的局部变量  
                  alert(a);  
               }  
  
                  alert(a)  
  
                   fn2()  
               }  
  
               //alert(a);  
  
               fn();  
     3.当声明一个全局的变量或者函数的时候，在window这个对象下，也会创建一个同名的属性，对应的值为变量的值  
       注意：a.如果是全局的变量或者函数，可以使用window.属性访问,省略掉window。  
          b.内置全局全局的变量下的属性，可以在任何一个地方使用。  
     4.内置全局的对象：window document Math  内置全局的方法：parseInt parseFloat alert()  Number()....  
     5.作用域链：变量与函数的查找规则。  
          \*\*\*声明的变量，如果被调用，会先在他所在当前的作用域查找，如果查找出来就会直接打印出当前所在作用域的值；  
          如果在当前所在作用域查找不到，就会逐级向上查找，如果有对应的变量值就会被打印出来，如果直到顶层没有查找  
          到，就会报错。  
     6.预解析：在代码运行之前，会有一个预解析的过程，会把声明的变量和函数提前处理  
            把声明的变量和函数提前在 当前作用域 最开始处理。  
     7.闭包：函数嵌套函数 内层函数访问外层函数的变量，这就是闭包。  
                               
  
  
  
  
js单词的书写格式的释义  
getElementById:get--获取 element--元素 id--id的名字  
     注意："id名字"必须加上单引号或者双引号  
onclick:on+click 当被点击的时候  
onmouseover:鼠标移入的时候  
onmouseout:鼠标移出的时候  
window.onload 当页面加载完成后，执行的事件。  
alert（）警告框或提示框（把要警告的内容放在alert后面的括号中）  
getComputedStyle（元素）get--得到  Computed--计算后的/渲染后   Style--样式。  
     不用定义直接使用。返回值是一个对象。取得具体的样式写法：getComputedStyle（元素）.样式名  如：getComputedStyle（box）.width

     需要考虑兼容性问题。

十五.定时器：javaScript中系统提供的函数（方法）。

     分类：重复执行定时器  延迟执行定时器 。

     作用：每隔一段时间或者延迟一段时间执行一段指定的代码

    1.重复执行定时器：setInterval(要重复执行的函数,间隔时间);

       参数：1：要重复执行的函数，把要重复执行的代码放到这个函数中，等到指定的时间的时候，就会执行这个函数

             2：间隔时间，指定每次执行第一个参数的时间；用毫秒来表示

    2.停止定时器：clearInterval(要停止定时器的编号)。

    3.延迟定时器：setTimeout(延迟执行的函数,延迟时间)。

       参数：1. 延迟执行的函数,把要延迟执行的代码放到这个函数中，等时间到的时候，会执行这个函数

         2. 延迟时间，指定执行第一个参数的时间

\*\*什么时候使用定时器

    当需要隔一段时间重复执行一段代码时用 setInterval()

    当需要延迟一段时间执行一段代码时用 setTimeout()

注意：a.定时器的返回值是一个数字，是表示这个定时器被开启的编号，得到这个数字，以便于后边对定时器的管理。

      b.重复定时器中的this问题，定时器“中”的this指向的对象是window。所以如果要用this需要在定时器外定义一个变量存储this。

      c.当清除的是一个还未开启的定时器，不会报错.写入数字也可以停止定时器，

    但是不建议使用，因为页面中多个地方开启定时器的时候，顺序不同，编号不一样

      d.当开启一个定时器的时候，系统会给这个定时器分配一个编号。

      e.延迟定时器执行时，当到指定的时间，会执行这个函数，只会执行一次。

      f.延迟定时器不需要清除。

十六.日期对象：用于处理日期和时间。

    日期对象：new Date();

    参数：1.无参：会以当前的时间为依据，创建一个时间对象。

          2.new Date(2016,4,1,9,48,12)参数：年 月 日 时 分 秒

          3. new Date('June 10,2013 12:12:12')字符串 外国人常用的日期格式

          4.new Date(时间戳)\*\*时间戳：从1970年1月1日0时0分0秒到指定日期的毫秒数。

    注意：时间对象不具备动态的特性，要实时的获取。

          月份是从0开始的。0-11月，分别对应1-12月

 获取时间对象的写法：getFullYear()获取年；getMonth()获取月份（月份从0开始计算）；getDate()获取日；

            getDay()获取星期（星期从周日开始，从0开始计算）；getHours()获取小时；getMinutes()获取分钟

            getSeconds()获取秒钟；getMilliseconds()获取毫秒；getTime() 获取1970-01-01 00:00:00到目前日期对象的时间差（单位:毫秒）

十七.操作字符串方法

    1.字符串：使用一对单引号或双引号包含起来的由0个或多个字符组成的串.

       注意：字符串的长度只能读取，不能设置。

             返回值类型 : 表示函数调用后的返回值的类型

         对象.方法名称 : 表示函数的调用方式

         参数列表 : 表示调用函数的时候传入的参数，[] 表示可选参数

    2.str.charAt():方法返回字符串中指定位置的字符。参数：默认为0；

       注意：a.在高版本浏览器下直接使用str[indedx]就可以找到指定位置的字符

           但是在IE6中为undefined

             b.如果参数小于0或者大于字符串.length-1会返回一个空字符串

    3.charCodeAt（动态方法）:返回指定索引处字符的 Unicode 数值。语法：number/str.charCodeAt(index)

                返回值：返回指定索引处字符的 Unicode 数值（number类型）

       参数：index为一个大于等于 0，小于字符串长度的整数。如果不是一个数值，则默认为 0

       注意：a.注意：如果传入的参数小于0||大于str.length-1 返回值为NaN

             b.如果为number时，index的值的范围是根据number的字符串长度来确定的。

         c.只有一个参数。

    4.String.fromCharCode：（静态方法）将指定 Unicode 数值转成正常的字符 。

       注意：参数：一个或者多个，用逗号隔开

    5.indexOf():返回指定字符在整个字符串首次出现的位置(返回指定值在调用该方法的字符串中首次出现的位置)

               语法：number string.indexOf(searchValue [, fromIndex])

               参数：searchValue 一个字符串表示被查找的值(要查找的字符)

      fromIndex  开始查找的位置，默认值为0，如果传入的数字为负数，那么自动转成0

               返回值：如果存在返回位置，则返回数字；如果不存在返回位置或者传入的第二个参数大于字符串的（长度-1），也返回-1

    6.lastIndexOf()：返回指定值在调用该方法的字符串中最后出现的位置（从后往前找，返回是从前往后数）

               语法：number string.lastIndexOf(searchValue [, fromIndex])

                参数：searchValue 一个字符串表示被查找的值

      fromIndex 开始查找的位置，默认值为str.length-1

                返回值为：返回数字。a.如果存在返回位置，如果不存在返回-1

                  b.如果第二个参数为负数，那么转成0

                  c.如果第二个参数比str.length-1还长，会转成str.length-1

    7.slice():方法提取字符串中的一部分，并返回这个新的字符串。

              语法：string string.slice(beginSlice[, endSlice])

          参数：beginSlice 起始位置  endSlice 结束位置，默认为最后一个字符的位置

          返回值：返回截取后的字符串

      注意：a.如果写了结束位置，不包括结束位置的字符。

        b.也可以接受负数，-1为整个字符串的最后一个，别的负数就以此类推

        c.如果负数超过的字符串的长度值，这个时候返回空字符串。

        d.如果传入的数字，超过了字符串的length，返回整个字符串

        e.起始位置小于结束位置才能截取到，不然为空

    8.split():通过一个指定的字符（串）把原字符串分割成一个数组。语法：array string.split([separator] [, limit])

          参数：separator 分割符  limit 指定最多分割的数量，默认为全部(保留几个)

          返回值：返回一个数组，当没有分割符的时候，整个字符串将作为一个整体保存到数组。一般配合split用的是\*数组\*的join()方法。

          join:以链接符为依据，将数组中的字符串链接，返回一个完整的字符串

          注意：a.除了空字符串切割空字符串为空数组以外，别的都能切割出来。

            b.可以用空字符把里面每个元素分割成数组中的元素。

    9.substring():截取指定位置之间的字符串。语法：string string.substring(indexStart [, indexEnd])

          参数：indexStart 截取的起始位置    indexEnd 截取的结束位置，默认为最后一个字符的位置，不包括结束位置。

          返回值：返回截取后的字符串。

          注意：如果第一个参数比第二个参数大，那么它会偷偷地调换位置，然后再截取（如果第一个参数为负数，自动转成0）

    10.substr(): 截取指定起始位置和长度的子字符串。语法：string string.substr(start [, length])。

             参数:start 截取的起始位置

              length  截取的字符串长度，默认为起始位置到最后一个字符的字符长度

             返回值：返回截取后的字符串。

             注意：如果截取的个数超过了余下的长度，那么能截几个截几个。

十八.操作数组方法：一堆数据的组合。数组的构造函数     Array

    创建数组的方法：1.使用new的方式来创建数组。var arr=new Array().

                   2.直接定义一个数组  var arr=[元素]

        注意：如果使用new创建数组时，括号内直接填写一个数字时，会认为初始化这个数组的长度。

    1.Array.length  获取数组的长度，数据的个数（可读可写）。

        注意：a.数组长度设置的最小长度为0.

              b.改变数组的长度，不建议使用，使用时需注意浏览器兼容问题。

    2.Array.push()  添加一个或多个元素到数组的末尾

        语法：number Array.push(element1, ..., elementN)

        参数：element 可以是任意类型的一个或多个元素。

        返回值：添加元素后数组的长度

        注意：Array.push() 会改变原来的数组。

    3.Array.unshift():  添加一个或多个元素到数组的头部

        语法：Array.unshift(element1, ..., elementN)

        参数：element 可以是任意类型的一个或多个元素。

        返回值：添加元素后数组的长度

        注意：Array.unshift() 会改变原来的数组。

    4.Array.pop(): 删除数组最后一个元素

        语法：Array.pop();

        返回值：被删除的那个元素。

        注意：a.Array.pop()括号内不需要填任何参数 b.会改变原来的数组

    4.Array.shift(): 删除数组第一个元素

        语法：Array.shift();

        返回值：被删除的那个元素。

        注意：a.Array.shift()括号内不需要填任何参数 b.会改变原来的数组

    5.Array.concat():将传入的数组或非数组值与原数组合并,组成一个新的数组并返回。

        语法：array.concat(value1, value2, ..., valueN)

        参数：需要与原数组合并的数组或非数组值。不传入参数时，会复制原来的数组。

        返回值：合并后的新数组（产生新的数组）。

    6.Array.reverse()：颠倒数组中元素的位置（只是颠倒位置，不会按大小排序）

        语法：Array.reverse()，没有任何参数，不接受任何参数。

        返回值：颠倒后的数组，不会产生新数组。

    7.Array.join():将数组中的所有元素连接成一个字符串

        语法：arr.join([separator = ','])

        参数：无参数，默认是以逗号连接的

              可选，用于指定连接每个数组元素的分隔符。分隔符会被转成字符串类型；

              如果省略的话，默认为一个逗号。如果 seprator 是一个空字符串，那么数组中的所有元素将被直接连接。

        返回值：是一个新拼接的字符串。

    8.Array.slice():截取数组中一部分，并返回这个新的数组。

        语法：arr.slice([begin[，end]])

        参数：一个起始位置，一个结束位置。

        返回值：截取后的新的数组，产生一个新数组。

        注意：a.不写任何参数时，会从头截到尾。

              b.写一个参数时，从指定位置开始，截取到最后。

              c.写两个参数时，不包括结束位置的元素。

    9.Array.splice():删除替换数组的元素，以此修改数组的内容。

        语法：array.splice(start, deleteCount[, item1[, item2[, ...]]])

        参数：start  起始位置   deleteCount  删除长度    item  添加的内容

        返回值：被修改后的一个新数组（会改变原来的数组）

    10.Array.sort():对数组的元素进行排序。

        语法：arr.sort([compareFunction])

        参数：1.无参数时，默认的排序是从小到大，是以字符串的unicode编码排序的

              2.参数为函数 排序是根据函数的返回值来交换位置的，不接受负数参数。

            返回值 > 0 会交换位置

            返回值 <=0 不交换位置

        返回值：排序后的新数组，不会产生新数组，但会改变原来的数组。

    11.for循环可以遍历数组，for in用来遍历对象和数组。

        注意：for in 遍历数组时不能从指定位置开始遍历。

        for(var attr in obj){}  for in 语法

十九.操作数字方法：

    1.Math.ceil()  对数进行向上取整。 参数：为一个数字

        语法：Math.ceil(x)   返回值：返回一个数字，返回的数字为x向上取整后的值（数字类型）。

    2.Math.floor()  对数进行向下取整   参数：为一个数字

        语法：Math.floor(x)   返回值：返回一个数字，返回的数字为x向下取整后的值（数字类型）。

    3.Math.round() 对数进行四舍五入   参数：为一个数字

        语法：Math.round(x)   返回值：返回x四舍五入后的整数（数字类型）

    4.Math.abs 取一个或多个数字的绝对值。  参数：为一个或多个数字。

        语法：Math.abs(x)  返回值：返回x的绝对值（数字类型）

        注意：Math.abs()与abs()返回值不一样。Math.abs()返回值为一个数字，abs()返回值为一个数组。

    5.Math.random() 返回0-1之间的随机数，不包含1

         x - y之间的随机数  Math.round(Math.random()\*(y-x)) +x