**前言**

因为公司最近项目比较忙，没那么多空余的事件写文章了，所以这篇文章晚了几天发布。但是这也没什么关系，不过该来的，总是会来的。  
好了，其他的不多说的，大家在开发的时候应该知道，有很多常见的实例操作。比如数组去重，关键词高亮，打乱数组等。这些操作，代码一般不会很多，实现的逻辑也不会很难，下面的代码，我解释就不解释太多了，打上注释，相信大家就会懂了。但是，用的地方会比较，如果项目有哪个地方需要用，如果重复写的话，就是代码沉余，开发效率也不用，复用基本就是复制粘贴！这样是一个很不好的习惯，大家可以考虑一下把一些常见的操作封装成函数，调用的时候，直接调用就好！  
源码都放在github上了，大家想以后以后有什么修改或者增加的，欢迎大家来star一下ec-do。

**字符串操作**

**去除字符串空格**

|  |
| --- |
| //去除空格  type 1-所有空格  2-前后空格  3-前空格 4-后空格  function trim(str,type){      switch (type){          case 1:return str.replace(/\s+/g,"");          case 2:return str.replace(/(^\s\*)|(\s\*$)/g, "");          case 3:return str.replace(/(^\s\*)/g, "");          case 4:return str.replace(/(\s\*$)/g, "");          default:return str;      }  } |

**字母大小写切换**

|  |
| --- |
| /\*type  1:首字母大写  2：首页母小写  3：大小写转换  4：全部大写  5：全部小写   \* \*/  //changeCase('asdasd',1)  //Asdasd  function changeCase(str,type)  {      function ToggleCase(str) {          var itemText = ""          str.split("").forEach(              function (item) {                  if (/^([a-z]+)/.test(item)) {                      itemText += item.toUpperCase();                  }                  else if (/^([A-Z]+)/.test(item)) {                      itemText += item.toLowerCase();                  }                  else{                      itemText += item;                  }              });          return itemText;      }        switch (type) {          case 1:              return str.replace(/^(\w)(\w+)/, function (v, v1, v2) {                  return v1.toUpperCase() + v2.toLowerCase();              });          case 2:              return str.replace(/^(\w)(\w+)/, function (v, v1, v2) {                  return v1.toLowerCase() + v2.toUpperCase();              });          case 3:              return ToggleCase(str);          case 4:              return str.toUpperCase();          case 5:              return str.toLowerCase();          default:              return str;      }  } |

**字符串循环复制**

|  |
| --- |
| //repeatStr(str->字符串, count->次数)  //repeatStr('123',3)  //"123123123"  function repeatStr(str, count) {      var text = '';      for (var i = 0; i < count; i++) {          text += str;      }      return text;  } |

**字符串替换**

|  |
| --- |
| //字符串替换(字符串,要替换的字符,替换成什么)  function replaceAll(str,AFindText,ARepText){  　　　raRegExp = new RegExp(AFindText,"g");  　　　return str.replace(raRegExp,ARepText);  } |

**替换\***

|  |
| --- |
| //replaceStr(字符串,字符格式, 替换方式,替换的字符（默认\*）)  function replaceStr(str, regArr, type,ARepText) {      var regtext = '', Reg = null,replaceText=ARepText||'\*';      //replaceStr('18819322663',[3,5,3],0)      //188\*\*\*\*\*663      //repeatStr是在上面定义过的（字符串循环复制），大家注意哦      if (regArr.length === 3 && type === 0) {          regtext = '(\\w{' + regArr[0] + '})\\w{' + regArr[1] + '}(\\w{' + regArr[2] + '})'          Reg = new RegExp(regtext);          var replaceCount = repeatStr(replaceText, regArr[1]);          return str.replace(Reg, '$1' + replaceCount + '$2')      }      //replaceStr('asdasdasdaa',[3,5,3],1)      //\*\*\*asdas\*\*\*      else if (regArr.length === 3 && type === 1) {          regtext = '\\w{' + regArr[0] + '}(\\w{' + regArr[1] + '})\\w{' + regArr[2] + '}'          Reg = new RegExp(regtext);          var replaceCount1 = repeatSte(replaceText, regArr[0]);          var replaceCount2 = repeatSte(replaceText, regArr[2]);          return str.replace(Reg, replaceCount1 + '$1' + replaceCount2)      }      //replaceStr('1asd88465asdwqe3',[5],0)      //\*\*\*\*\*8465asdwqe3      else if (regArr.length === 1 && type == 0) {          regtext = '(^\\w{' + regArr[0] +  '})'          Reg = new RegExp(regtext);          var replaceCount = repeatSte(replaceText, regArr[0]);          return str.replace(Reg, replaceCount)      }      //replaceStr('1asd88465asdwqe3',[5],1,'+')      //"1asd88465as+++++"      else if (regArr.length === 1 && type == 1) {          regtext = '(\\w{' + regArr[0] +  '}$)'          Reg = new RegExp(regtext);          var replaceCount = repeatSte(replaceText, regArr[0]);          return str.replace(Reg, replaceCount)      }  } |

**检测字符串**

|  |
| --- |
| //checkType('165226226326','phone')  //false  //大家可以根据需要扩展  function checkType (str, type) {      switch (type) {          case 'email':              return /^[\w-]+(\.[\w-]+)\*@[\w-]+(\.[\w-]+)+$/.test(str);          case 'phone':              return /^1[3|4|5|7|8][0-9]{9}$/.test(str);          case 'tel':              return /^(0\d{2,3}-\d{7,8})(-\d{1,4})?$/.test(str);          case 'number':              return /^[0-9]$/.test(str);          case 'english':              return /^[a-zA-Z]+$/.test(str);          case 'chinese':              return /^[\u4E00-\u9FA5]+$/.test(str);          case 'lower':              return /^[a-z]+$/.test(str);          case 'upper':              return /^[A-Z]+$/.test(str);          default :              return true;      }  } |

**检测密码强度**

|  |
| --- |
| //checkPwd('12asdASAD')  //3(强度等级为3)  function checkPwd(str) {      var nowLv = 0;      if (str.length < 6) {          return nowLv      }      ;      if (/[0-9]/.test(str)) {          nowLv++      }      ;      if (/[a-z]/.test(str)) {          nowLv++      }      ;      if (/[A-Z]/.test(str)) {          nowLv++      }      ;      if (/[\.|-|\_]/.test(str)) {          nowLv++      }      ;      return nowLv;  } |

**随机码（toString详解）**

|  |
| --- |
| //count取值范围0-36    //randomNumber(10)  //"2584316588472575"    //randomNumber(14)  //"9b405070dd00122640c192caab84537"    //Math.random().toString(36).substring(2);  //"83vhdx10rmjkyb9"  function randomNumber(count){     return Math.random().toString(count).substring(2);  } |

**查找字符串**

可能标题会有点误导，下面我就简单说明一个需求，在字符串'sad44654blog5a1sd67as9dablog4s5d16zxc4sdweasjkblogwqepaskdkblogahseiuadbhjcibloguyeajzxkcabloguyiwezxc967'中找出'blog'的出现次数。代码如下

|  |
| --- |
| function countStr (str,strSplit){      return str.split(strSplit).length-1  }  var strTest='sad44654blog5a1sd67as9dablog4s5d16zxc4sdweasjkblogwqepaskdkblogahseiuadbhjcibloguyeajzxkcabloguyiwezxc967'  //countStr(strTest,'blog')  //6 |

**数组操作**

**数组去重**

这个方法是在太多了，我之前写的文章(js数组操作--使用迭代方法替代for循环,js关键词变色，数组打乱，数组去重的实现和封装)也有提到，我今天这里就写一种之前没用过的方法。

|  |
| --- |
| //ES6新增的Set数据结构，类似于数组，但是里面的元素都是唯一的 ，其构造函数可以接受一个数组作为参数  //let arr=[1,2,1,2,6,3,5,69,66,7,2,1,4,3,6,8,9663,8]  //let set = new Set(array);  //{1,2,6,3,5,69,66,7,4,8,9663}  //ES6中Array新增了一个静态方法from，可以把类似数组的对象转换为数组  //Array.from(set)  //[1,2,6,3,5,69,66,7,4,8,9663]  function removeRepeatArray(arr){      return Array.from(new Set(arr))  } |

**数组顺序打乱**

|  |
| --- |
| function upsetArr(arr){      return arr.sort(function(){ return Math.random() - 0.5});  } |

**数组最大值最小值**

|  |
| --- |
| //这一块的封装，主要是针对数字类型的数组  function maxArr(arr){      return Math.max.apply(null,arr);  }  function minArr(arr){      return Math.min.apply(null,arr);  } |

**数组求和，平均值**

|  |
| --- |
| //这一块的封装，主要是针对数字类型的数组  //求和  function sumArr(arr){   var sumText=0;   for(var i=0,len=arr.length;i<len;i++){   sumText+=arr[i];   }   return sumText  }  //平均值,小数点可能会有很多位，这里不做处理，处理了使用就不灵活了！  function covArr(arr){   var sumText=sumArr(arr);   var covText=sumText/length;   return covText  } |

**从数组中随机获取元素**

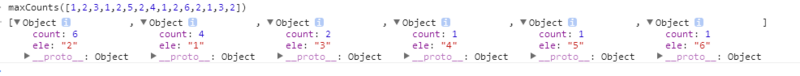
|  |
| --- |
| //randomOne([1,2,3,6,8,5,4,2,6])  //2  //randomOne([1,2,3,6,8,5,4,2,6])  //8  //randomOne([1,2,3,6,8,5,4,2,6])  //8  //randomOne([1,2,3,6,8,5,4,2,6])  //1  function randomOne(arr) {      return arr[Math.floor(Math.random() \* arr.length)];  } |

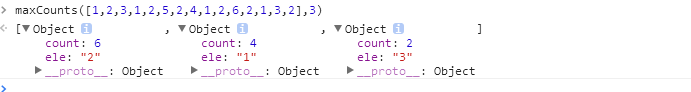
**返回数组（字符串）一个元素出现的次数**

|  |
| --- |
| //getEleCount('asd56+asdasdwqe','a')  //3  //getEleCount([1,2,3,4,5,66,77,22,55,22],22)  //2  function getEleCount (obj, ele) {      var num = 0;      for (var i = 0, len = obj.length; i < len; i++) {          if (ele == obj[i]) {              num++;          }      }      return num;  } |

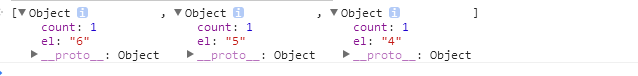
**返回数组（字符串）出现最多的几次元素和出现次数**

|  |
| --- |
| //arr, rank->长度，默认为数组长度，ranktype，排序方式，默认降序  function getCount(arr, rank，ranktype){      var obj = {}, k, arr1 = []      //记录每一元素出现的次数      for (var i = 0, len = arr.length; i < len; i++) {          k = arr[i];          if (obj[k]) {              obj[k]++;          }          else {              obj[k] = 1;          }      }      //保存结果{el-'元素'，count-出现次数}      for (var o in obj) {          arr1.push({el: o, count: obj[o]});      }      //排序（降序）      arr1.sort(function (n1, n2) {          return n2.count - n1.count      });      //如果ranktype为1，则为升序，反转数组      if(ranktype===1){          arr1=arr1.reverse();      }      var rank1 = rank || arr1.length;      return arr1.slice(0,rank1);  } |

getCount([1,2,3,1,2,5,2,4,1,2,6,2,1,3,2])  
默认情况，返回所有元素出现的次数  


getCount([1,2,3,1,2,5,2,4,1,2,6,2,1,3,2],3)  
传参（rank=3），只返回出现次数排序前三的  


getCount([1,2,3,1,2,5,2,4,1,2,6,2,1,3,2],null,1)  
传参（ranktype=1,rank=null），升序返回所有元素出现次数  


getCount([1,2,3,1,2,5,2,4,1,2,6,2,1,3,2],3,1)  
传参（rank=3，ranktype=1），只返回出现次数排序（升序）前三的  


**得到n1-n2下标的数组**

|  |
| --- |
| //getArrayNum([0,1,2,3,4,5,6,7,8,9],5,9)  //[5, 6, 7, 8, 9]    //getArrayNum([0,1,2,3,4,5,6,7,8,9],2) 不传第二个参数,默认返回从n1到数组结束的元素  //[2, 3, 4, 5, 6, 7, 8, 9]  function getArrayNum(arr,n1,n2){      var arr1=[],len=n2||arr.length-1;      for(var i=n1;i<=len;i++){          arr1.push(arr[i])      }      return arr1;  } |

**筛选数组**

|  |
| --- |
| /删除值为'val'的数组元素  //removeArrayForValue(['test','test1','test2','test','aaa'],'test','%')  //["aaa"]   带有'test'的都删除    //removeArrayForValue(['test','test1','test2','test','aaa'],'test')  //["test1", "test2", "aaa"]  //数组元素的值全等于'test'才被删除  function removeArrayForValue(arr,val,type){      arr.filter(function(item){return type==='%'?item.indexOf(val)!==-1:item!==val})  } |

**基础DOM操作**

这个部分代码其实参考jquery的一些函数写法，唯一区别就是调用不用，参数一样.  
比如下面的栗子

|  |
| --- |
| //设置对象内容  jquery：$('#xxx').html('hello world');  现在：html(document.getElementById('xxx'),'hello world')  //获取对象内容  jquery：$('#xxx').html();  现在：html(document.getElementById('xxx')) |

**检测对象是否有哪个类名**

|  |
| --- |
| function hasClass(obj,classStr){      var arr=obj.className.split(/\s+/); //这个正则表达式是因为class可以有多个,判断是否包含      return (arr.indexOf(classStr)==-1)?false:true;  }, |

**添加类名**

|  |
| --- |
| function addClass(obj,classStr){      if (!this.hasClass(obj,classStr)){obj.className += " " + classStr};  }, |

**删除类名**

|  |
| --- |
| function removeClass(obj,classStr){      if (this.hasClass(obj,classStr)) {          var reg = new RegExp('(\\s|^)' + classStr + '(\\s|$)');          obj.className = obj.className.replace(reg, '');      }  }, |

**替换类名("被替换的类名","替换的类名")**

|  |
| --- |
| function replaceClass(obj,newName,oldName) {      removeClass(obj,oldName);      addClass(obj,newName);  } |

**获取兄弟节点**

|  |
| --- |
| function siblings(obj){      var a=[];//定义一个数组，用来存o的兄弟元素      var p=obj.previousSibling;      while(p){//先取o的哥哥们 判断有没有上一个哥哥元素，如果有则往下执行 p表示previousSibling          if(p.nodeType===1){          a.push(p);          }          p=p.previousSibling//最后把上一个节点赋给p      }      a.reverse()//把顺序反转一下 这样元素的顺序就是按先后的了      var n=obj.nextSibling;//再取o的弟弟      while(n){//判断有没有下一个弟弟结点 n是nextSibling的意思          if(n.nodeType===1){              a.push(n);          }          n=n.nextSibling;      }      return a;  }, |

**设置样式**

|  |
| --- |
| function css(obj,json){      for(var attr in json){          obj.style[attr]=json[attr];      }  } |

**设置文本内容**

|  |
| --- |
| function html(obj){      if(arguments.length==0){          return this.innerHTML;      }      else if(arguments.length==1){          this.innerHTML=arguments[0];      }  } |

**显示隐藏**

|  |
| --- |
| function show(obj){      obj.style.display="";  }  function hide(obj){      obj.style.display="none";  } |

**其他操作**

**cookie**

|  |
| --- |
| //cookie  //设置cookie  function setCookie(name,value,iDay){   var oDate=new Date();   oDate.setDate(oDate.getDate()+iDay);   document.cookie=name+'='+value+';expires='+oDate;  }  //获取cookie  function getCookie(name){   var arr=document.cookie.split('; ');   for(var i=0;i<arr.length;i++){   var arr2=arr[i].split('=');   if(arr2[0]==name)   {   return arr2[1];   }   }   return '';  }  //删除cookie  function removeCookie(name){   setCookie(name,1,-1);  } |

**清除对象中值为空的属性**

|  |
| --- |
| //filterParams({a:"",b:null,c:"010",d:123})  //Object {c: "010", d: 123}  function filterParams(obj){      let \_newPar = {};      for (let key in obj) {          if ((obj[key] === 0 || obj[key]) && obj[key].toString().replace(/(^\s\*)|(\s\*$)/g, '') !== '') {              \_newPar[key] = obj[key];          }      }      return \_newPar;  } |

**现金额大写转换函数**

|  |
| --- |
| //upDigit(168752632)  //"人民币壹亿陆仟捌佰柒拾伍万贰仟陆佰叁拾贰元整"  //upDigit(1682)  //"人民币壹仟陆佰捌拾贰元整"  //upDigit(-1693)  //"欠人民币壹仟陆佰玖拾叁元整"  function upDigit(n)  {      var fraction = ['角', '分','厘'];      var digit = ['零', '壹', '贰', '叁', '肆', '伍', '陆', '柒', '捌', '玖'];      var unit = [ ['元', '万', '亿'], ['', '拾', '佰', '仟']  ];      var head = n < 0? '欠人民币': '人民币';      n = Math.abs(n);      var s = '';      for (var i = 0; i < fraction.length; i++)      {          s += (digit[Math.floor(n \* 10 \* Math.pow(10, i)) % 10] + fraction[i]).replace(/零./, '');      }      s = s || '整';      n = Math.floor(n);      for (var i = 0; i < unit[0].length && n > 0; i++)      {          var p = '';          for (var j = 0; j < unit[1].length && n > 0; j++)          {              p = digit[n % 10] + unit[1][j] + p;              n = Math.floor(n / 10);          }          //s = p.replace(/(零.)\*零$/, '').replace(/^$/, '零')+ unit[0][i] + s;          s = p+ unit[0][i] + s;      }      return head + s.replace(/(零.)\*零元/, '元').replace(/(零.)+/g, '零').replace(/^整$/, '零元整');  } |

**获取，设置url参数**

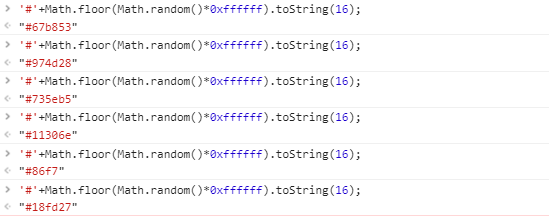
|  |
| --- |
| //获取url参数  //getUrlPrmt('segmentfault.com/write?draftId=122000011938')  //Object{draftId: "122000011938"}  function getUrlPrmt(url) {      url = url ? url : window.location.href;      let \_pa = url.substring(url.indexOf('?') + 1), \_arrS = \_pa.split('&'), \_rs = {};      for (let i = 0, \_len = \_arrS.length; i < \_len; i++) {          let pos = \_arrS[i].indexOf('=');          if (pos == -1) {              continue;          }          let name = \_arrS[i].substring(0, pos), value = window.decodeURIComponent(\_arrS[i].substring(pos + 1));          \_rs[name] = value;      }      return \_rs;  }    //设置url参数  //setUrlPrmt({'a':1,'b':2})  //a=1&b=2  function setUrlPrmt(obj) {      let \_rs = [];      for (let p in obj) {          if (obj[p] != null && obj[p] != '') {              \_rs.push(p + '=' + obj[p])          }      }      return \_rs.join('&');  } |

**随机返回一个范围的数字**

|  |
| --- |
| function randomNumber(n1,n2){      //randomNumber(5,10)      //返回5-10的随机整数，包括5，10      if(arguments.length===2){          return Math.round(n1+Math.random()\*(n2-n1));      }      //randomNumber(10)      //返回0-10的随机整数，包括0，10      else if(arguments.length===1){          return Math.round(Math.random()\*n1)      }      //randomNumber()      //返回0-255的随机整数，包括0，255      else{          return Math.round(Math.random()\*255)      }  } |

**随进产生颜色**

|  |
| --- |
| function randomColor(){      //randomNumber是上面定义的函数      //写法1      return 'rgb(' + randomNumber(255) + ',' + randomNumber(255) + ',' + randomNumber(255) + ')';        //写法2      return '#'+Math.random().toString(16).substring(2).substr(0,6);        //写法3      var color='#';      for(var i=0;i<6;i++){          color+='0123456789abcdef'[randomNumber(15)];      }      return color;  }  //这种写法，偶尔会有问题。大家得注意哦  //Math.floor(Math.random()\*0xffffff).toString(16); |



**Date日期时间部分**

|  |
| --- |
| //到某一个时间的倒计时  //getEndTime('2017/7/22 16:0:0')  //"剩余时间6天 2小时 28 分钟20 秒"  function getEndTime(endTime){      var startDate=new Date();  //开始时间，当前时间      var endDate=new Date(endTime); //结束时间，需传入时间参数      var t=endDate.getTime()-startDate.getTime();  //时间差的毫秒数      var d=0,h=0,m=0,s=0;      if(t>=0){        d=Math.floor(t/1000/3600/24);        h=Math.floor(t/1000/60/60%24);        m=Math.floor(t/1000/60%60);        s=Math.floor(t/1000%60);      }      return "剩余时间"+d+"天 "+h+"小时 "+m+" 分钟"+s+" 秒";  } |

**适配rem**

这个适配的方法很多，我就写我自己用的方法。大家也可以去我回答过得一个问题那里看更详细的说明！移动端适配问题

|  |
| --- |
| function getFontSize(){      var doc=document,win=window;      var docEl = doc.documentElement,      resizeEvt = 'orientationchange' in window ? 'orientationchange' : 'resize',      recalc = function () {          var clientWidth = docEl.clientWidth;          if (!clientWidth) return;          //如果屏幕大于750（750是根据我效果图设置的，具体数值参考效果图），就设置clientWidth=750，防止font-size会超过100px          if(clientWidth>750){clientWidth=750}          //设置根元素font-size大小          docEl.style.fontSize = 100 \* (clientWidth / 750) + 'px';      };      //屏幕大小改变，或者横竖屏切换时，触发函数      win.addEventListener(resizeEvt, recalc, false);      //文档加载完成时，触发函数      doc.addEventListener('DOMContentLoaded', recalc, false);  }  //使用方式很简单，比如效果图上，有张图片。宽高都是100px;  //样式写法就是  img{      width:1rem;      height:1rem;  }  //这样的设置，比如在屏幕宽度大于等于750px设备上，1rem=100px；图片显示就是宽高都是100px  //比如在iphone6(屏幕宽度：375)上，375/750\*100=50px;就是1rem=50px;图片显示就是宽高都是50px; |

**封装成形**

写了这么多的操作，小伙伴应该发现了一问题，全局函数太多了

|  |
| --- |
| //这样的话，封装了几个操作，就增加了几个全局函数，污染了全局变量，在开发中应该尽量避免全局变量。不说别的，如果一个项目，几个人开发，很有可能会造成命名的冲突。    function setUrlPrmt(obj){..}  function getUrlPrmt(url){..}  function upsetArr(obj){..}      //所以，建议的封装姿势是  var myJS={      setUrlPrmt:function(obj){..},      getUrlPrmt:function(url){..},      upsetArr:function(arr){..},  }  这样就算别人也写这样的函数,也不会造成冲突。全局变量也只有一个，加上别人也不会很多！  var otherJS={      setUrlPrmt:function(obj){..},      getUrlPrmt:function(url){..},      upsetArr:function(arr){..},  }      //最后，封装的效果是  var myJS={      //去除字符串空格      trim:function(obj){..},      //字母大小写切换      changeCase:function(obj){..},      //字符串循环复制      repeatStr:function(str){..},      .....  }    //如果是es6的模块化开发，大家也可以  let myJS={      //去除字符串空格      trim:function(obj){..},      //字母大小写切换      changeCase:function(obj){..},      //字符串循环复制      repeatStr:function(str){..},      .....  }  //暴露模块，里面的方式大家也可以用es6方式实现，代码至少会少一点！  export {myJS} |

可能有小伙伴会有疑问，这样封装，调用有点麻烦，为什么不直接在原型上面封装，调用方便。比如下面的栗子！

|  |
| --- |
| String.prototype.trim=function(type){      switch (type){          case 1:return this.replace(/\s+/g,"");          case 2:return this.replace(/(^\s\*)|(\s\*$)/g, "");          case 3:return this.replace(/(^\s\*)/g, "");          case 4:return this.replace(/(\s\*$)/g, "");          default:return this;      }  }  //'  12345 6 8 96  '.trim(1)  //"123456896"  //比这样trim('  12345 6 8 96  ',1)调用方便。  //但是，这样是不推荐的做法，这样就污染了原生对象String,别人创建的String也会被污染，造成不必要的开销。  //更可怕的是，万一自己命名的跟原生的方法重名了，就被覆盖原来的方法了  //String.prototype.substr=function(){console.log('asdasd')}  //'asdasdwe46546'.substr()  //asdasd  //substr方法有什么作用，大家应该知道，不知道的可以去w3c看下 |

所以在原生对象原型的修改很不推荐！至少很多的公司禁止这样操作！

**小结**

这篇文章，写了很久了，几个小时了，因为我写这篇文章，我也是重新改我以前代码的，因为我以前写的代码，功能一样，代码比较多，现在是边想边改边写，还要自己测试（之前的代码for循环很多，现在有很多简洁的写法代替）。加上最近公司比较忙，所以这一篇文章也是花了几天才整理完成。  
源码都放在github上了，大家想以后以后有什么修改或者增加的，欢迎大家来star一下ec-do。  
我自己封装这个，并不是我有造轮子的习惯，而是：  
1，都是一些常用，但是零散的小实例，网上基本没有插件。  
2，因为零散的小实例，涉及到的有字符串，数组，对象等类型，就算找到插件，在项目引入的很有可能不止一个插件。  
3.都是简单的代码，封装也不难。维护也简单。  
废话就不多说了，上面的只是我自己在开发中常用到，希望能帮到小伙伴们，最理想就是这篇文章能起到一个抛砖引玉的作用，就是说，如果觉得还有什么操作是常用的，或者觉得我哪里写得不好的，也欢迎指出，让大家相互帮助，相互学习。

原文地址：<https://segmentfault.com/a/1190000010225928#articleHeader39>