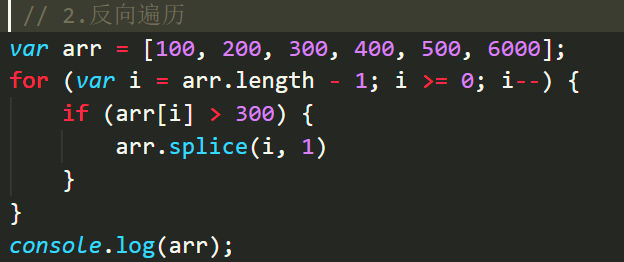
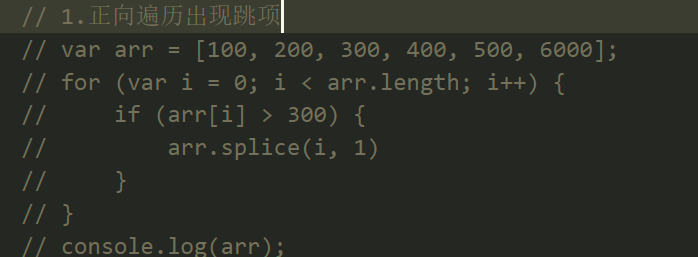
1. 关于反向遍历的问题：方向遍历的俩个应用（数组翻转和数组中删除某些元素）正向遍历删除条件成立时，设置i- -也可以解决

情景：遍历数组，满足要求的从数组中删除，如果正向遍历的话，只能当某一个满足的要求的元素被删除出后，后面元素的索引会自动减1，所会出现跳项的问题，此时只能用反向遍历来

反向遍历因为从数组的最后一个元素开始检索，删除后并不不会影响数组的索引值：

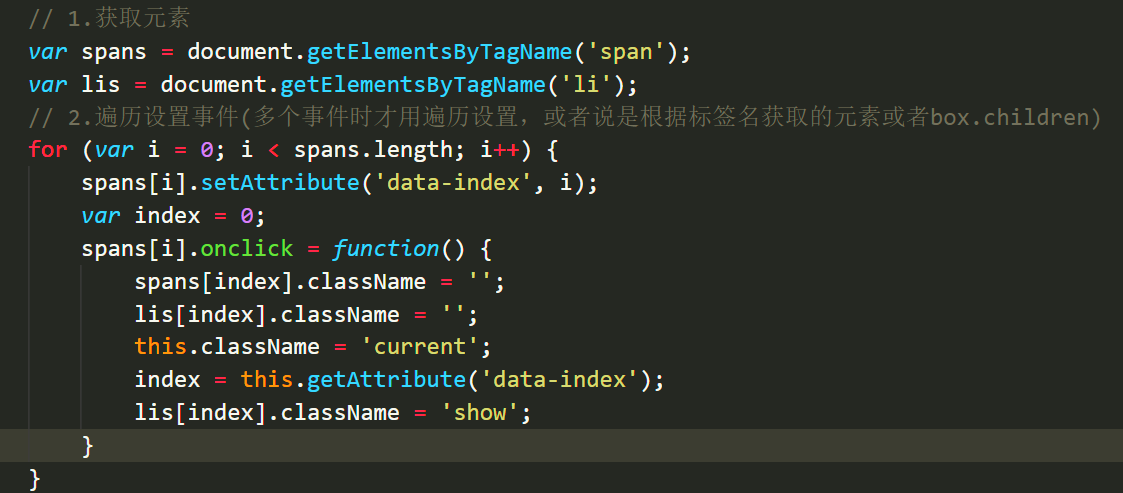
1. tab栏切换效果
2. tab 栏切换都是根据被点击的元素的索引,获取到对应索引的下方盒子显示;

先遍历设置事件，遍历设置事件的同时遍历设置每个事件源的自定义行内属性(setAttribute(‘data-index’,i))，以便获取索引值

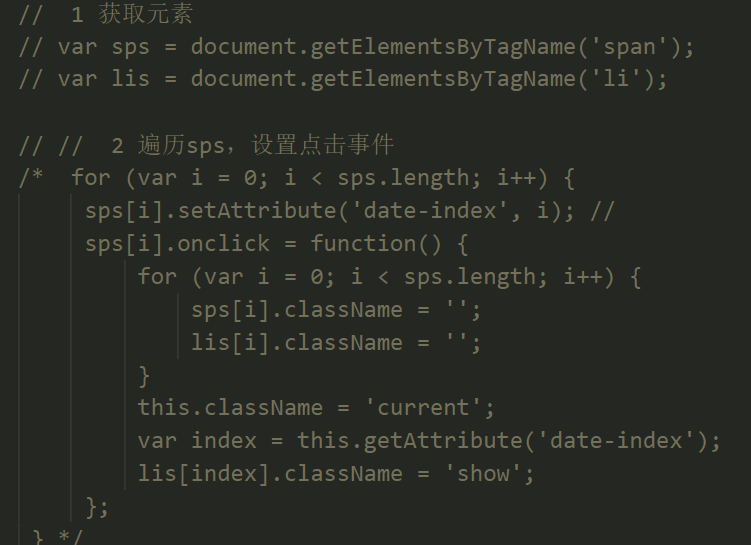
当点击某一个事件时：

1先清除上一次点击事件的效果（第一次点击的话就默认清除索引值为0，即数组中第一个点击事件的效果 设置全局变量 var index = 0;） (实在不行就for循环清除所有的效果吧)

1. 然后设置当前点击事件的效果
2. 记录当前点击事件的索引值 getAttribute()

­­­­

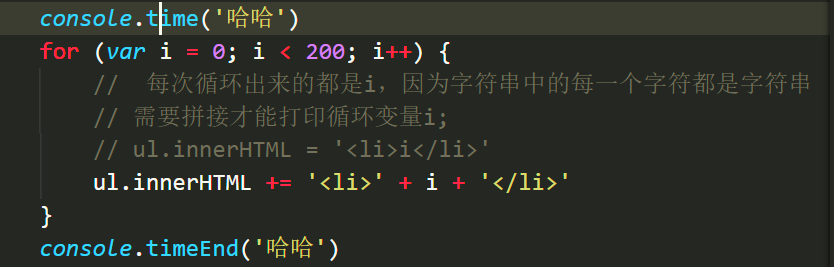
清除所有：



3.indexOf()方法 str.indexOf（‘’）； 在数组中有兼容性问题，IE9下不支持。在string的方法中没有兼容性问题



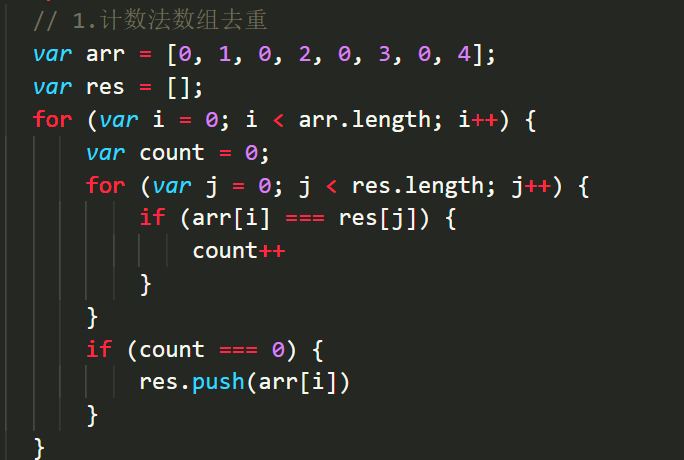
4.关于字符串拼接的问题：



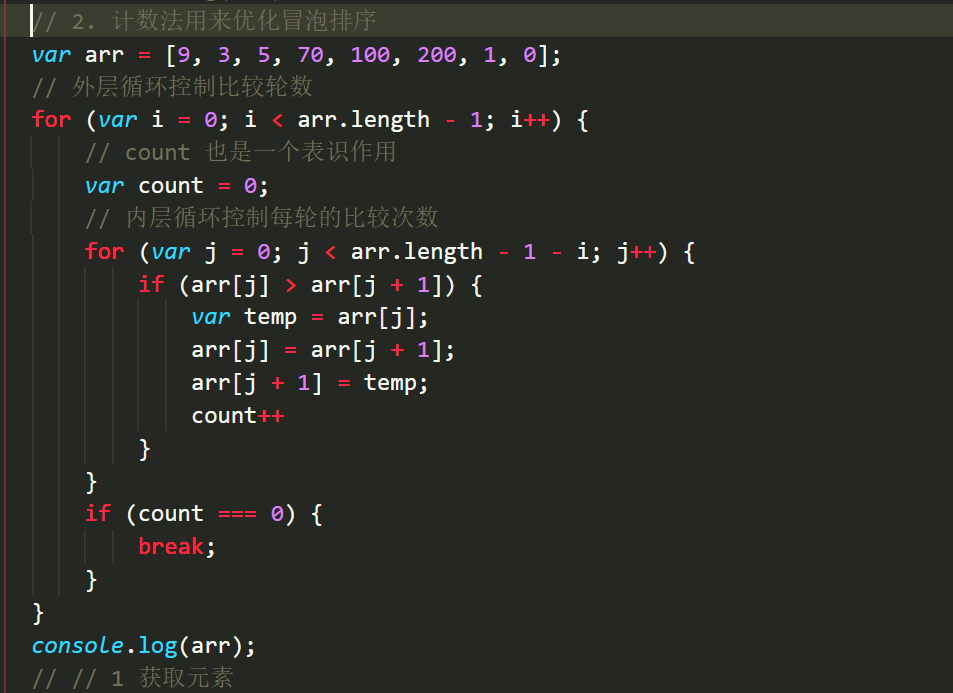
5.计数法的应用：（按钮的反选，数组去重，和冒泡排序的优化）



5.2 数组去重用计数法来来作为标识



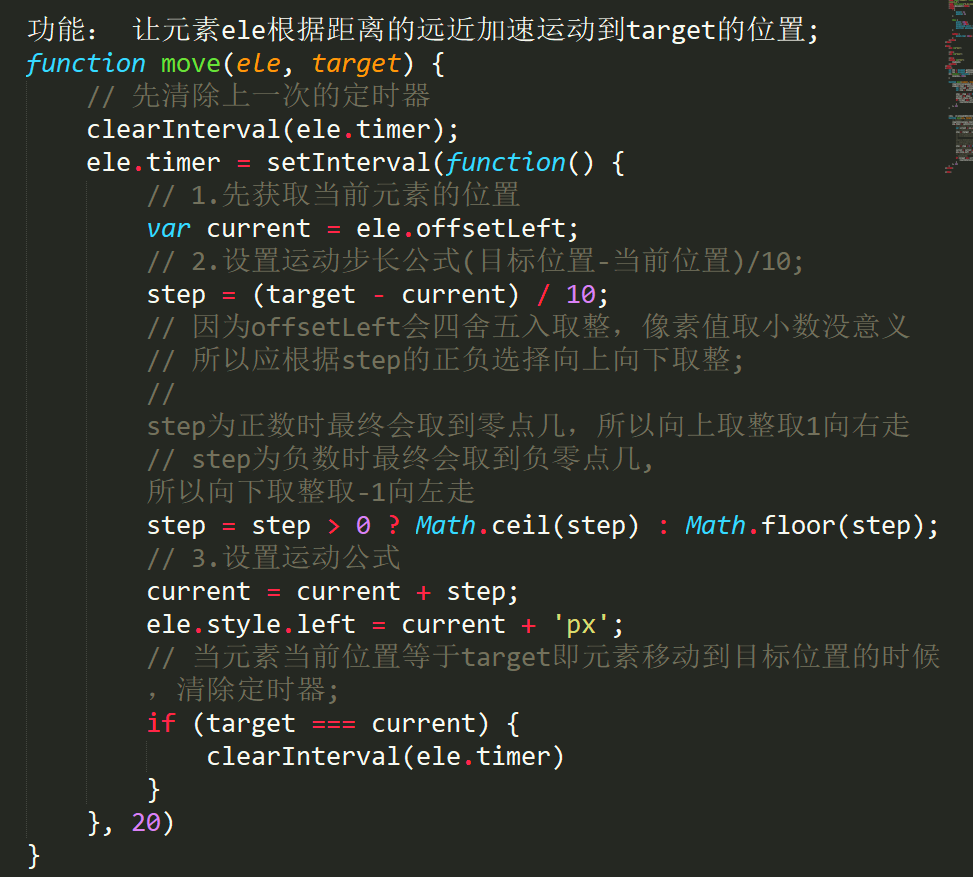
5.3计数法用来优化冒泡排序



# 6. JS减速运动函数封装



7. JS匀速运动对象封装



8. 回调函数，模拟sort（）排序方法（length不要写错,js为弱类型语言，解析时遇到不认识的单词会自动跳过）

Fn为mySort()的形参，function(a.b){return a-b}这个函数体为实参; a b 为形参，对应的实参为this[j],和this[j+1];



9.原生js实现深拷贝：

// 1.浅拷贝：只能拷贝一层。在多层结构中,对于内层结构的拷贝

// 只能拷贝内层结构的指针,拷贝前后俩组数据有一定的关联性,+

// 达不到深拷贝的效果。

// 2.深拷贝：拷贝前后俩组数据类型没有任何关联,修改任意一组数据

// 另一组数据不会改变。



9.检测某个字符串中是否存在某些元素，存在返回索引值。

