1、写一个通用的事件侦听器函数markyun.**Event** = { *// 参数： 操作的元素,事件名称 ,事件处理程序* addEvent: **function** (element, type, handler) {  
 **if** (element.addEventListener) {  
 *//事件类型、需要执行的函数、是否捕捉* element.addEventListener(type, handler, **false**);  
 } **else if** (element.attachEvent) {  
 element.attachEvent(**'on'** + type, **function** () {  
 handler.call(element);  
 });  
 } **else** {  
 element[**'on'** + type] = handler;  
 }  
 },  
 *// 移除事件* removeEvent: **function** (element, type, handler) {  
 **if** (element.removeEventListener) {  
 element.removeEventListener(type, handler, **false**);  
 } **else if** (element.datachEvent) {  
 element.detachEvent(**'on'** + type, handler);  
 } **else** {  
 element[**'on'** + type] = **null**;  
 }  
 },  
 *// 阻止事件 (主要是事件冒泡，因为IE不支持事件捕获)* stopPropagation: **function** (ev) {  
 **if** (ev.stopPropagation) {  
 ev.stopPropagation();  
 } **else** {  
 ev.**cancelBubble** = **true**;  
 }  
 },  
 *// 取消事件的默认行为* preventDefault: **function** (event) {  
 **if** (event.preventDefault) {  
 event.preventDefault();  
 } **else** {  
 event.**returnValue** = **false**;  
 }  
 },

*// 获取事件目标* getTarget: **function** (event) {  
 **return** event.**target** || event.**srcElement**;  
 }  
 }  
  
  
  
2、如何判断一个对象是否为数组  
<**script**>  
 **function** *isArray*(arg) {  
 **if** (**typeof** arg === **'object'**) {  
 **return** Object.**prototype**.toString.call(arg) === **'[object Array]'**;  
 }  
 **return false**;  
 }  
</**script**>  
  
3、冒泡排序  
每次比较相邻的两个数，如果后一个比前一个小，换位置  
<**script**>  
 **var *arr*** = [3, 1, 4, 6, 5, 7, 2];  
 **function** *bubbleSort*(arr) {  
 **for** (**var** i = 0; i < arr.**length** - 1; i++) {  
 **for**(**var** j = 0; j < arr.**length** - 1; j++) {  
 **if**(arr[j + 1] < arr[j]) {  
 **var** temp;  
 temp = arr[j];  
 arr[j] = arr[j + 1];  
 arr[j + 1] = temp;  
 }  
 }  
 }  
 **return** arr;  
 }  
 **console**.log(*bubbleSort*(***arr***));  
</**script**>

4、快速排序  
采用二分法，取出中间数，数组每次和中间数比较，小的放到左边，大的放到右边  
<**script**>  
 **var *arr*** = [3, 1, 4, 6, 5, 7, 2];  
 **function** *quickSort*(arr) {  
 **if**(arr.**length** == 0) {  
 **return** []; *// 返回空数组* }  
 **var** cIndex = **Math**.floor(arr.**length** / 2);  
 **var** c = arr.splice(cIndex, 1);  
 **var** l = [];  
 **var** r = [];  
 **for** (**var** i = 0; i < arr.**length**; i++) {  
 **if**(arr[i] < c) {  
 l.push(arr[i]);  
 } **else** {  
 r.push(arr[i]);  
 }  
 }  
 **return** *quickSort*(l).concat(c, *quickSort*(r));  
 }  
 **console**.log(*quickSort*(***arr***));  
</**script**>  
  
5、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  
<**script**>  
 **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"**));  
</**script**>

6、bind的用法，以及如何实现bind的函数和需要注意的点  
bind的作用与call和apply相同，区别是call和apply是立即调用函数，而bind是返回了一个函数，需要调用的时候再执行。  
一个简单的bind函数实现如下  
<**script**>  
 **Function**.**prototype**.bind = **function**(ctx) {  
 **var** fn = **this**;  
 **return function**() {  
 fn.apply(ctx, arguments);  
 };  
 };  
</**script**>