# webApi第四天--事件详解

（事件这部分的主要内容 事件的绑定形式 事件对象 事件流的三个阶段（事件冒泡 事件目标 事件捕获）键盘事件 事件的默认行为 事件拖拽）

1. 事件详解--事件的多种绑定方式
   1. 传统绑定方式 obj.on+事件名称 = function(){}

这种方式 同一个对象同一个事件 绑定多个事件处理函数 会发生后者覆盖前者

问题:我们往往在实际开发中，需要分工协作，这个时候在交叉工作区域极可能会将他人的功能进行覆盖

b) 另一种绑定方式 obj.addEventListener(evType,evFn,false)

这种方式 绑定的时候不会覆盖

参数：1 绑定对象 2 绑定的事件处理函数 3 是否捕获 false为冒泡

在IE内核的浏览器中 有一个绑定方式 obj.attachEvent(on+evType,evFn)

两者的区别：

1 事件绑定是否加on

addEventListener 事件类型不需要加on

而attachEvent需要加on

2 事件执行顺序

addEventListener()正序执行

attachEvent() 在标准Ie下正序执行 在Ie6 7 8倒序执行

3 this指向

addEventListener()函数内部的this指向没问题指向当前对象

attachEvent() 它内部的this指向有问题 不是指向当前对象 指向window

补充了一个知识点：call方法

在函数下面有一个call() 可以手动更改函数内部的this指向

例如： fn.call(btn);// 你可以将fn函数里面的this主动改成btn

参数:fn.call() 它如果不写参数 相当于fn()

如果写参数：

第一个参数： 你需要更改的this指向的对象

从第二个参数开始：后面可以写实参

Function fn(a,b) {

Console.log(a+b)

//This.stye.……

}

oBox.onclick = function(){

//Fn(12,15)

Fn.call(btn,12,15)

};

绑定事件的兼容性函数封装

// 这是一个绑定方式的兼容性函数

function bind(obj,evType,evFn) {

// 根据浏览器能力进行检测 如果识别支持addEventListener 就直接使用这个绑定方式

// 如果不支持这个方法 则按照后面的方式进行绑定

obj.handle = function(){

evFn.call(obj);

}

if(obj.addEventListener){

// 标准浏览器走这个绑定

obj.addEventListener(evType,evFn,false);

obj.handle = evFn;

}else if(obj.attachEvent) {

//IE6 7 8 走这个绑定方式

obj.attachEvent("on"+evType,obj.handle);

} else {

// 以上方法都不支持的很老的浏览器 走这个方法

obj["on"+evType] = evFn;

}

}

c) 事件的解绑方法

Obj.removeEventListener(evType,evFn,false)

Obj.detachEvent(“on”+evType,evFn)

解绑方式兼容性函数的封装

// 这是封装了一个解绑的函数

function unbind(obj,evType,evFn) {

if(obj.removeEventListener) {

obj.removeEventListener(evType,evFn,false);

}else if(obj.detachEvent) {

obj.detachEvent("on"+evType,evFn);

}else {

obj["on"+evType] = null;

}

}

1. 事件对象event：
   1. 什么是事件对象：当一个对象触发一个事件的时候 ，会把当前触发的事件有关的详细信息，存储到一个地方----这个地方就是event事件对象里面，在需要的时候可以使用里面的信息
   2. 比如：飞机黑匣子： event
   3. 事件对象怎么获取
      1. 在IE/chrome 浏览器中 事件对象event 是属于全局内置对象 属于window下面的属性 window.event
      2. 标准浏览器都支持（IE9以及以上 谷歌 火狐） 事件对象到底在哪儿？在事件触发的时候，作为事件处理函数的第一个参数传入进来的
   4. 事件对象处理兼容
      1. ev = ev || event;
   5. 使用一下事件对象下的属性
      1. ev.clientX 当前事件发生的时候 鼠标相对于可视区域的x坐标
      2. ev.clientY当前事件发生的时候 鼠标相对于可视区域的y坐标
      3. 补充一个事件：

mousemove事件：鼠标移动事件 ：触发频率 和鼠标移动距离（像素无关），和移动速度有关

在很短的一个单位时间内，它会判断这一次和上一次位置是否发生改变，如果改变了，则触发

案例： div跟着鼠标移动

原理：