一：事件类型

如何绑定一个事件(3种方法)

1: dom.onclick = function(){}

优点： 兼容性好，

缺点：

同一个元素的同一个事件上只能绑定一个事件处理函数

如果写了（ 同一个元素的同一个事件上绑定了两个事件处理函数）

属性赋值，后面的会覆盖前面的；

这种写法相当于

写在元素的行内样式

```
<div style="width: 200px;" onclick="console.log(1)"></div>
```

直接写执行的内容，不用写function(){}

2:

```
dom.addEventListener("click", function() {}, false);
dom.addEventListener("事件类型", 事件处理函数, true/false)
```

给一个对象的的一个事件绑定多个处理函数（不同的函数），并且按照绑定的顺序去执行

给一个对象的的一个事件绑定多个处理相同的函数，即引用地址是一样的函数，只会执行一次

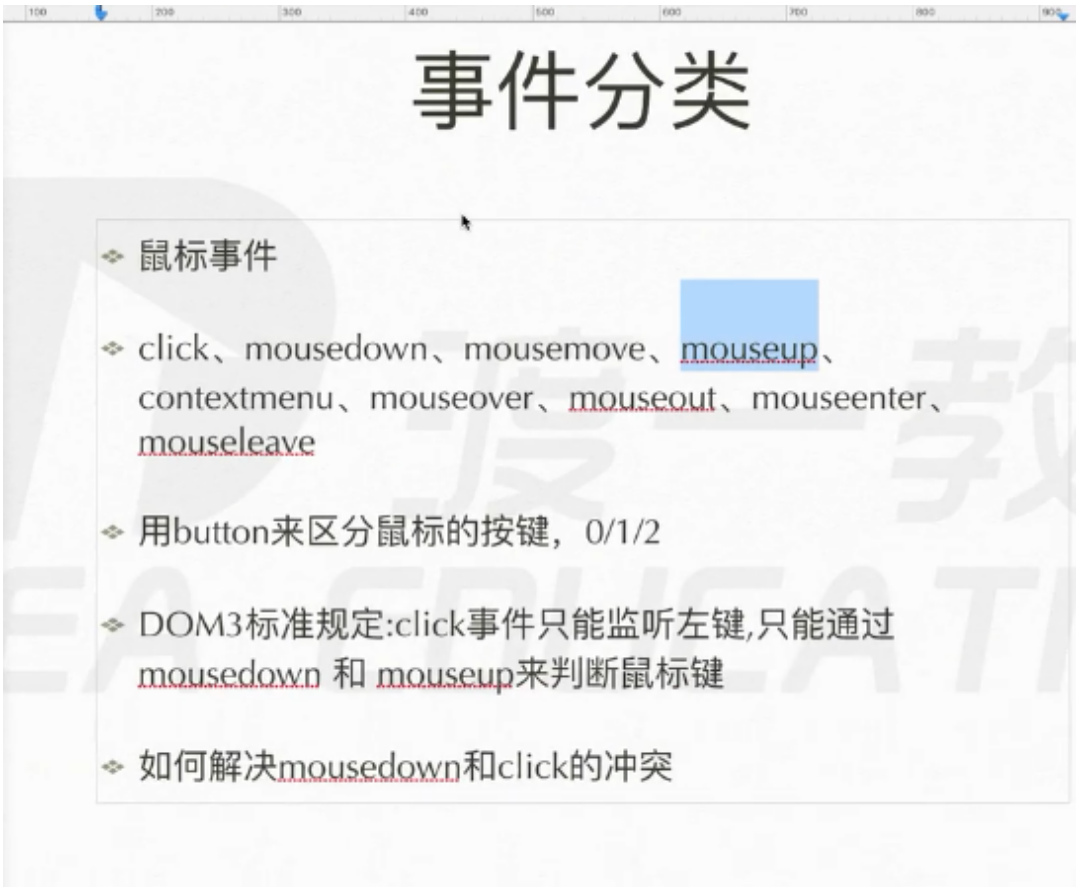
3：ie独有的

dom.attachEvent("on"+事件类型，处理函数)

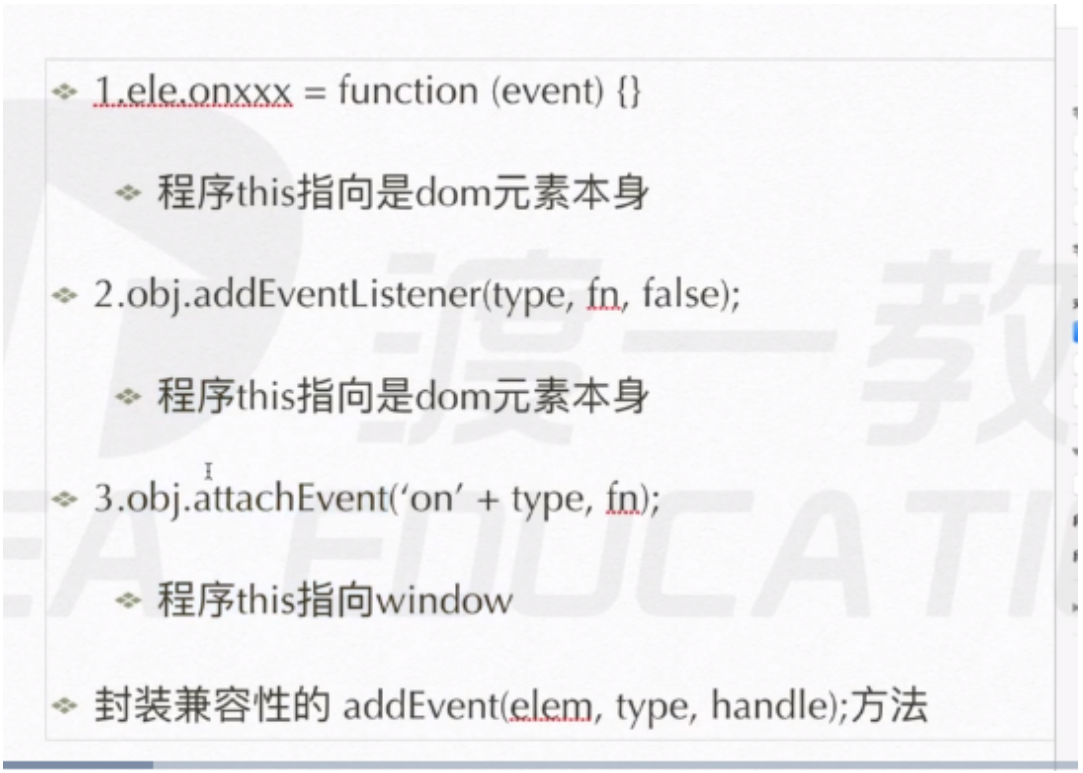
给一个对象的的一个事件绑定多个处理函数（不同的函数），并且按照绑定的顺序去执行

给一个对象的的一个事件绑定多个处理相同的函数，即引用地址是一样的函数，也会执行多次

二：



三：事件中的this指向



ie中 的事件绑定中的this指向window

```
dom.attachEvent("onclick",function(){
console.log(this) ----》window
})
```

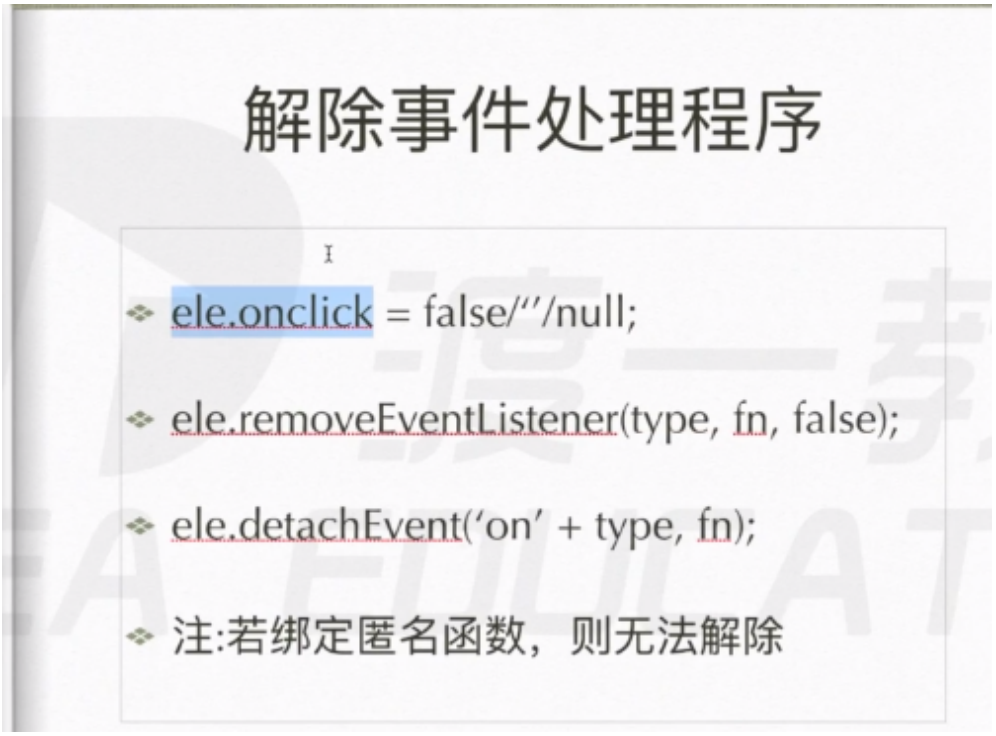
处理办法，this就会指向当前函数

```
    li.attachEvent("onclick",function(){
        handle.call(li);
    })
    function handle(){
     this.
    }
```

四：　//封装不同浏览器的事件处理的方法

```
function addEvent(elem, type, handle) {
    if (elem.addEventListener) {
        elem.addEventListener(type, handle, false);
    } else if (elem.attachEvent) {
        elem.attachEvent('on' + type, function () {
            handle.call(elem);
        })
    } else {
        //用on的方法绑定事件
        elem["on" + type] = handle
    }
}
```

五：解除事件处理程序



1：

```
//只能执行一次的事件
dom.onclick = function () {
    console.log("a");
    this.onclick = null;    //写在代码里面 ，执行完就失效
}
```

2:

```
dom.addEventListener('click', function () {
  console.log("a");
}, false)
dom.removeEventListener('click',,false)
```

这么写的话，即匿名函数， 这个事件是清除不了的，

想要清除，处理函数写成函数的引用