## 如何给事件处理函数传参数却不立即执行

document.getElementById("wrong").onclick = wrongFun;

function wrongFun(){

*console*.log(*arguments*[0]);

}

先来看看这段代码，点击按钮后，你认为他会打印出什么？

document.getElementById("wrong").onclick = wrongFun();

function wrongFun(){

*console*.log(*arguments*[0]);

}

再来看看这段代码，点击按钮后，他又会打印出什么？

第一段代码会输出以下内容：（点击之后输出）

MouseEvent {isTrusted: true, screenX: 68, screenY: 119, clientX: 68, clientY: 17, …}

第二段代码输出一下内容：（还没点击呢就输出了）

undefined

理解并区分这两个代码很简单，看函数wrongFun后的括号就知道，没有括号的就是click事件的处理函数，有括号的就是当场执行了函数，然后将返回的内容作为click事件的处理。

这里我想说并不是有没有括号，而是在于函数的第一个参数到底是什么，我们不难发现带括号的函数的第一个参数并没有得到赋值（严谨说得到了默认赋值undefined），而函数本身也在最开始即被执行，相当于一个普通的无参立即执行函数，本身和onclick无关。因此，返回undefined并不奇怪。

但是对于没有括号的函数，我们都知道他是一个指针，只不过在这里和onclick有了绑定的关系，函数本身作为onclick的事件处理函数。js默认将event对象作为第一个参数传入了这个事件处理函数之中。因此在发生点击事件后，会在控制台将event对象打印出来。

这样问题就出现了，如果我想要参数中默认第一个位置含有event对象，又想拥有其他的参数该怎么办？

如果想要传入参数，可能会想到加一个括号，第一个手机event，第二个是需要的参数即可。像是

document.getElementById("wrong").onclick = wrongFun(event,a);

这样问题就回到了刚才说的情况。函数作为一个普通的函数立即执行，event和a都只是一个普通的形参。

如果不加括号就不能传参，因此如果存在封装需求，比如想要让不同的按钮点击出来发生同样的通知行为，但是通知的内容却不一样，总不能给每一个函数都写一个具体的内容，这样函数的冗余很大，极其不合适。

我想到了一个办法，就是巧妙地利用闭包封装消息内容。

既然，立即执行的函数的返回内容会作为onclick的处理存在，那就只需要让函数返回一个处理函数即可，形成一个闭包，将需要传入的不同参数写入外部函数中，然后通过内部函数调用外部函数的参数。

就像这样：

var msg = "弹出的消息";

document.getElementById("wrong").onclick = rightFun(msg);

function rightFun(*mess*){

return function(*e*){

// e = window.event || arguments.callee.caller.arguments[0];

// 上面的可有可无，因为内部函数本身并无其他参数，第一个参数默认为event，

// 但是如果内部函数有其他参数，那么第一个参数必须显示的声明用来存放event，

// 或者解除上面的哪行代码的注释。

*console*.log(e); // MouseEvent {isTrusted: true, screenX: 68, screenY: 122, clientX: 68, clientY: 20, …}

*console*.log(mess); // 弹出的消息

}

}

逻辑相同的部分只需要封装在内部的闭包之中即可。

闭包的函数也可写成这样，这样更能凸显处理函数会自动为第一个参数赋值为event对象的特点

function rightFun(*mess*){

return function(*a*,*b*,*c*){ // a自动成为event

// e = window.event || arguments.callee.caller.arguments[0];

*console*.log(a); // MouseEvent {isTrusted: true, screenX: 68, screenY: 122, clientX: 68, clientY: 20, …}

*console*.log(mess); // 弹出的消息

}

}

或者

function rightFun(*mess*){

return function(){

// e = window.event || arguments.callee.caller.arguments[0];

// 上面的可有可无，因为内部函数本身并无其他参数，第一个参数默认为event，

// 但是如果内部函数有其他参数，那么第一个参数必须显示的声明用来存放event，

// 或者解除上面的哪行代码的注释。

*console*.log(*arguments*[0]); // MouseEvent {isTrusted: true, screenX: 68, screenY: 122, clientX: 68, clientY: 20, …}

*console*.log(mess); // 弹出的消息

}

}

## 需要补充的部分

如果将Javascript的事件处理嵌入到html的代码里面，第一个参数就不会自动带入event（其实是window.event，window可省），而且也不会立即执行（毕竟是html环境）。

<button *id* = "right" *onclick* = "myFun(1,2)">正确的事件处理</button>

function myFun(*a*,*b*,*c*){

*console*.log(a); // 1

*console*.log(b); // 2

}

如果想要是用event，只需要将参数传入event对象即可。

<button *id* = "right" *onclick* = "myFun(1,event)">正确的事件处理</button> // 注意event直接传入即可，不需要一定放在第一个参数的位置。

function myFun(*a*,*b*,*c*){

*console*.log(a);

*console*.log(b);

}

或者是像之前说的一样，普遍的在函数内部获得event对象的方式。

function myFun(*a*,*b*,*c*){

e = window.event || *arguments*[0].callee.caller.arguments[0];

*console*.log(e);

*console*.log(a);

*console*.log(b);

}

## 总结如下

1. 将事件处理函数写成一个闭包的形式，可以灵活的使用作为参数的event对象，也可加入其它参数，只需要将雷同的逻辑封装在闭包中即可。
2. 事件处理函数会将第一个参数作为event。
3. 可以使用一下代码来普遍的获得event（推荐）。

e = window.event || *arguments*.callee.caller.arguments[0];

（4）HTML内部嵌入的事件处理函数，并不会将第一个参数默认作为event对象。需要显示的传入event对象，且不一定在第一个参数位置（网上有的说必须是第一个位置，我认为不然）。