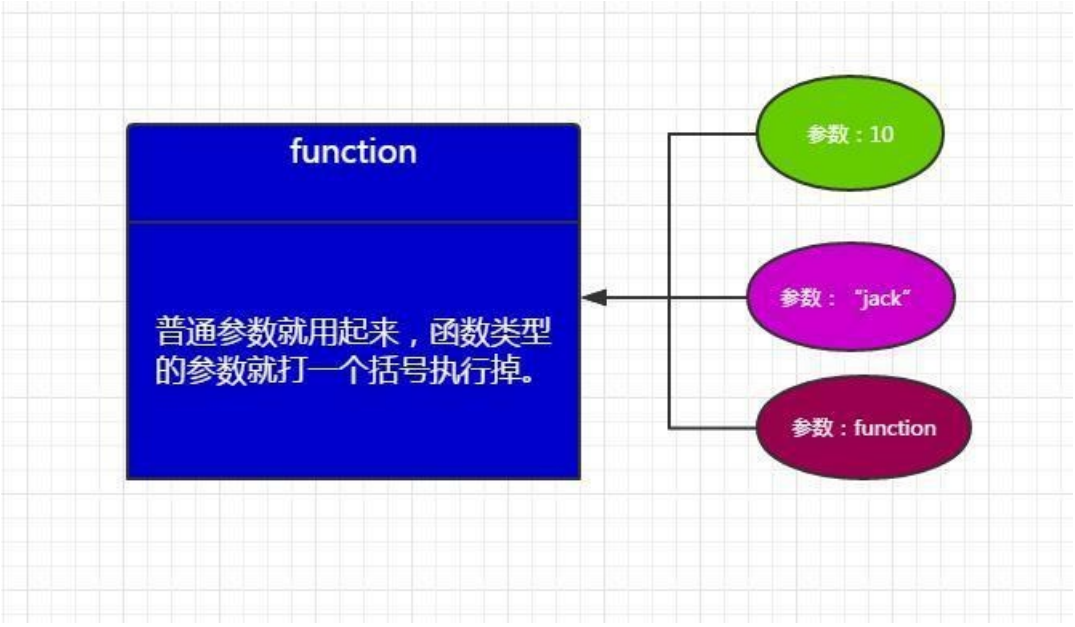
小兔JS教程（三）-- 彻底攻略JS回调函数



这一讲来谈谈回调函数。

其实一句话就能概括这个东西：

回调函数就是把一个函数当做参数，传入另一个函数中。传进去的目的仅仅是为了在某个时刻去执行它。

如果不执行，那么你传一个函数进去干嘛呢？

就比如说对弈下棋，如果你都不想赢，那么你为什么要下棋？当然了，如果你达到了某种至高无上的境界，参悟出一个“道”来，就不一样了。

所谓手中无剑，心中有剑。写了一个函数，我虽然没有去执行它，但是在我心中已经执行了。

在此我们先不谈那么高大上的境界，先说点俗的，你想想啊，你好不容易写了一个function，你不去执行执行它，那你为什么要写呢？

1.回调函数快速入门

先来个快速入门吧。

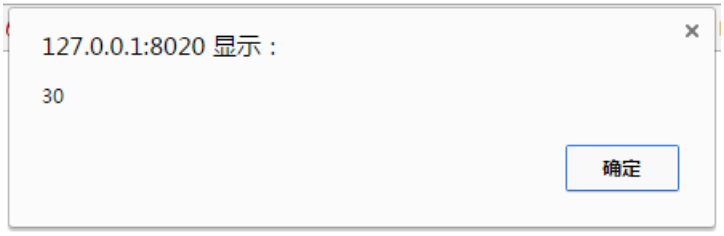
比如我有两个数字，分别为10和20，还定义了两个函数，一个是做加法，一个是做减法。

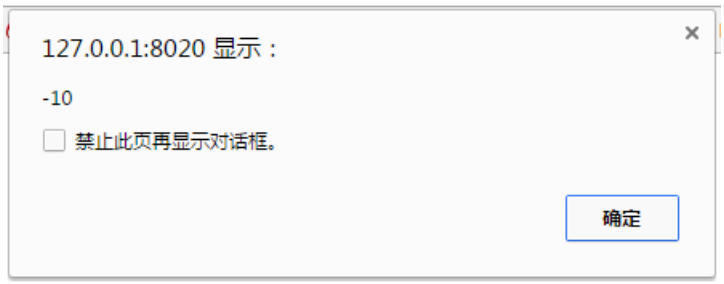
```
var num1 = 10;
var num2 = 20;

//加法
function add(num1,num2){
    alert(num1 + num2);
}

//减法
function minus(num1,num2){
    alert(num1 - num2);
}
```

这样应该没问题，你肯定能看懂。

运行一下：

```
add(num1,num2);
minus(num1,num2);
```





好的，没问题。现在来思考，有没有什么办法，我创造一个函数，同时具备了加法和减法的功能呢？

当然有了，我大不了传一个标志位flag,如果是0，代表加法，如果是1，就代表减法。

像这样：

```
function addOrMinus(flag,num1,num2){
    if(flag == 0){
        alert(num1 + num2);
    }

    if(flag == 1){
        alert(num1 - num2);
    }
}
```

这样固然是可以的，但是还有个问题，如果用户突然说，我不想要加减法了，我要做乘除法，那又该怎么办？用户的需求是千变万化的，如果我们把函数里面的内容写死，那么就显得非常不灵活。这个时候，你就会想，有没有什么办法，让函数的功能变得不确定起来呢？

我们在刚才的例子中，是这样实现加减法的，即传入一个标志位flag，如果flag＝1，就做减法，如果flag＝2，就做加法。也就是说，加法和减法的逻辑已经实现在函数里面写好了，所以，一旦我们需要做乘法和除法，就不得不修改函数体，对不对。

那与其这样，我们为什么不能把具体的逻辑实现交给用户呢？你要做加法，你就给我传一个加法的逻辑进来，你要做减法，你就给我传一个减法的逻辑进来。这样不就好了？看代码：

```
function compute (a,b,callback){
    callback(a,b);
}

//加法
compute(10,12,function(num1,num2){
    alert(num1+num2);
});

//减法
compute(10,12,function(num1,num2){
    alert(num1-num2);
});

//乘法
compute(10,12,function(num1,num2){
    alert(num1*num2);
});

//除法
compute(10,12,function(num1,num2){
    alert(num1/num2);
});
```

callback就是回调函数的意思，，定义一个函数和定义一个其他变量都是差不多的。比如你定义一个数字：

```
var a = 100;
```

是不是这样做的，那么定义一个函数不就是：

```
var fun = function(){
    alert('你好! ');
}
```

不是一个意思吗，不知道我这样写你是不是好理解一点呢？

我们定义了一个变量a，它的值为100，那么如果我们使用这个a，是不存在什么执不执行的问题的，直接调用就OK了，这就是所谓的执行了一次右查询。

比如我写：alert(a),那么a的值就被我拿到了，并且使用了。

可函数的话呢，有点不一样。比如上面的例子，你总不可能这么写吧。

fun;

你觉得这样子会执行吗？肯定不会嘛，因为函数它必须要打一个括号才能执行啊！你不打括号的话它就执行不了的。

fun();

这样子写，它才会执行函数体里面的内容。

再回过头来看之前的例子：

```
function compute (a,b,callback){
    callback(a,b);
}
```

我定义了一个函数，叫做compute，它接受了三个参数，分别是a，b,callback。你不要觉得害怕，说callback是啥玩意啊，是不是一定要写callback呢？不是的啊，你不要多想了，callback只是为了让别人一看就知道是回调函数，这样显得更加语义化。实际上你写aaa,bbb,ccc都没有问题。他只是一个参数的名字啊。你叫阿猫阿狗都没事的。你信不信咯？它无非就只是一个变量的名字而已。

比如你写 var a = 10; 这个你肯定知道，我写a只是随便写的，写b、c、都可以，没有问题。那callback不也是一个意思吗？我之所以要这么啰嗦，是希望以后如果你看到别人js框架里面，或者某个API文档也写callback，你不要再害怕了，也不要再恐惧了，觉得哎呀好难，callback是什么东西？？它就是一个名字而已。

OK，回到这个函数。

```
function compute (a,b,callback){
    callback(a,b);
}
```

你说，我这样一写了之后，compute函数有没有被执行？对了，当然是没有。因为我没打括号嘛，作为一个函数，打了括号才会被执行。同样的，在compute的函数体里面，传进去的callback是不是打了括号，这么写的用意很明显，它就是为了在函数体里面把你传进来的callback函数执行掉。并且在执行的时候，把另外两个参数传给它。

compute函数承担了计算的任务，具体怎么计算，我不管，计算规则由你决定！也就是说，你给我一个回调函数callback，我不管三七二十一，帮你执行掉。就这么简单，回调函数就是这么简单，没有什么更加高深的东西在里面了。

2.回调函数应用场景

快速入门就到这里，接下来，我们来看几个典型的例子。

首先，我们在运用jQuery的时候，是不是总是写这样的代码：

```
$(function(){

});
```

很显然，这个就是回调函数，$本身就是一个函数的名字，没有道理不相信，我就问你，它是不是打了括号？我们把里面的function(){} 去掉：

$();

是不是就变成这样了？那好，我就想请问一下了，你见过除了函数之外的什么东西要打括号吗？有没有，就问你一句话，有还是没有？只有函数才能打括号啊，你写一个var a = 10; 能打括号吗？所以，对jQuery来说，它本身就是一个函数，这一点要明确。

接下来，是不是传了一个回调函数进去了？

```
$(function(){
```

```
});
```

没错，的确是的。我在函数体里面alert一下，它肯定会给我弹出一个提示来。为什么会这样呢，毫无疑问，jQuery肯定在里面把这个回调函数执行了。就比如：

```
var $ = function(callback){
    callback();
}
```

当然，这只是举一个例子，实际情况肯定要比这个复杂得多。

```
$(function(){

});
```

这个函数类似于window.onload。

接下来，我们来个例子：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <script src="http://libs.baidu.com/jquery/2.0.0/jquery.min.js"></script>
        <title></title>
        <script>

            $(function(){
                $("#box").on('click',function(){
                    alert();
                });
            });

        </script>
    </head>
    <body>
        <div style='padding: 50px;display:inline-block;background: blue;' id='box'></div>
    </body>
</html>
```


我在body区域画了一个蓝色的方块，并且通过jQuery的方式，给它绑定了一个点击事件。你可能会说，这么简单的代码我还能看不懂？我天天写这种代码呢！点击事件就是典型的回调函数应用，因为我哪里知道你点击之后要干什么啊？这当然要你自己决定啊。所以，你自己传一个回调函数进去。

```
$("#box").on('click',function(){
    alert();
});
```

现在，我给点击事件里面的回调函数加一个参数e。

```
$("#box").on('click',function(e){
    alert();
});
```

你可能会觉得奇怪，心想，纳尼，这个e是什么鬼？？然后马上又一想，我好像没有在哪里定义一个叫做e的变量啊。如果你这么想了，说明你还是没有理解啥叫函数。亲啊，这是一个函数啊，函数的参数是e，e只是一个名字啊，你写aaa,bbb,ccc都行的！

```
$("#box").on('click',function(aaa){
    alert();
});
```

一个名字而已嘛，而且，这里只是定义了一个函数：

```
function(e){
    alert();
}
```

我就问你，有没有打括号？如果你说有啊，（e）不是括号吗？如果你真的这么回答，那我就要哭了。。。回到正题，这里是不是还没有打括号？也就是说，我只是写了一个还未被执行的函数传进去了，这是一个回调函数。我知道，我传进去以后，你肯定会在某个地方打一个括号帮我执行的，就算不执行，它也肯定会把这个回调函数赋值给其他变量。这是第一点，第二点，我写的这个函数，还带了一个参数，参数的名字叫e。

OK，非常好。也就是说，我只管定义了一个有参数的函数，具体这个参数是啥，什么时候传进来，我不知道。这是由jQuery的on函数决定的。如果你还是不理解的话呢，我们先写一个简单点的例子：

```
function on(event,callback){
    if(event == 'click'){
        var e = "Hello World";
        callback(e);
    }
}

on('click',function(e){
    alert(e);
});
```

这样就很好理解了吧，对的，就是这么个意思。所以，当你看到别人写回调函数，还加了个参数e，最起码，不要再害怕了，它根本不高端，说来说去都是JavaScript的基础知识。所以有的人不重视基础，虽然也能完成开发任务，但却往往只是照猫画虎而已，真要出了点问题，就很难解决了。

让我们回到代码：

```
$("#box").on('click',function(e){
    alert();
});
```

这是jQuery给我们封装好的函数，专门用来绑定事件的。那么在这一讲的最后，我们就来模拟jQuery的写法，封装一个类似的功能吧！

页面上的div是这样的：

```
<div style='padding: 50px;display:inline-block;background: blue;' id='box'></div>
```

第一步，是不是要专门写一个函数，取到这个div。在js中，我们可以用document.getElementById的方式取到dom元素，现在我们将这个方法也单独封装起来。

```
var $ = function(id){
    return document.getElementById(id);
}
```

这样就行了，可是有个问题，这个函数返回的是一个dom对象，而标准的dom元素是没有绑定事件的方法的。所以，我们不能直接返回dom，而应该返回一个json。（json虽然是后面的内容，这里先提前用一下吧）

我返回一个 json，json的用处就大了，它是一个实实在在的对象，既有属性也有方法。在json中，属性和方法之间都是用逗号分隔的。

```
var $ = function(id){
    return {
        element : document.getElementById(id) ,
        on : function(event,callback){
            this.element['on' + event] = callback;
        }
    }
}
```

关于this呢，我会在以后的章节中详细讲的，现在先这么用。

如果用js的方法给dom元素添加一个点击事件，一般我们会这么写：

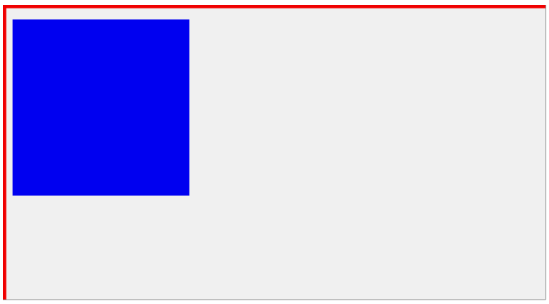
1.dom.onclick = function(){}

2.dom['onclick'] = function(){}

两种写法都可以哈，这样应该比较好理解了吧。我返回的json中，有一个函数叫做on，专门用来绑定事件的。比如现在我这么调用：

```
window.onload = function(){
    $('box').on('click',function(){
        $('box').element.style.background = 'green';
    });
}
```

我给div添加一个点击事件，效果就是变换一下背景色。



效果是有的，好的，那么这一讲先到这里了。

作业：

作业要求（自行编写 $ 函数和 operation 函数，实现以下的调用过程，不允许使用jQuery）：

注意“#”的处理，#代表id选择器。

```
$('#box').operation(function(){
//自行实现回调函数，将box的背景色变为pink
});
```

作业地址：http://www.xiaotublog.com/demo.html?path=homework/03/index