# callback 回调函数

以下内容借鉴老鸟的经验和知识，精髓的总结

* 一句话 对于以后研究node 和那些热门的前端框架 很有帮助。如果你看过这个文章 对于你来说 质的突变。

不bb那么多，翠花上酸菜

* 理解javascript中的回调函数(callback)，希望对你有所帮助。
* 在JavaScrip中，function是内置的类对象，也就是说它是一种类型的对象，和其它String、Array、Number、Object类的对象一样用于内置对象的管理。
* function实际上是一种对象，它可以“存储在变量中，通过参数传递给(别一个)函数(function)，在函数内部创建，从函数中返回结果值”。
* 函数式编程最主要的技术之一就是回调函数，实现回调函数就像传递一般的参数变量一样简单。

## 那么第一个问题来了，什么是回调？

* 回调函数被认为是一种高级函数，一种被作为参数传递给另一个函数(在这称作otherFunction)的高级函数。回调函数会在otherFunction内被调用(或执行)。
* 回调函数的本质是一种模式(一种解决常见问题的模式)，因此回调函数也被称为回调模式。

如下列子

var colors=[ 'red', 'yellow', 'blue', 'green'];  
colors.forEach(function(color,i){  
 console.log( i + 1 + '.' + color)// 1.red, 2.yellow, .blue, 4.green  
})

这里我们传递一个匿名函数给foreach方法，作为foreach的参数，传递了一个匿名的函数作为参数给另一个函数或方法

这里举一个 jquery 常用的回调函数

<button class="btn">btn</button>  
  
$(".btn").click(function() {  
 alert("clicked");  
});

同样这里 传递一个函数给click方法的参数，click方法将会调用（或执行）传递给他的回调函数。

## 讲了这么多，回调函数是怎么实现的呢？

回调函数作为变量一样使用，来作为另一个函数的参数，在另一个函数中作为返回的结果，在另一个函数中调用它。 这里我们作为参数传递一个回调函数给另一个函数时，我们只传递了函数的定义并没有在参数中执行它。

当函数（这里指的是调用或者执行函数）拥有了参数中定义的回调函数后，它可以在任何时候调用。 也说明了回调函数不是立即执行的，而是在包含函数的函数体内指定的位置回调它。

$(".btn").click(function() {//不执行匿名函数的参数  
 alert("clicked");  
});  
  
//匿名函数将延迟在click函数的函数体内被调用，即使没有名称，也可以被包含函数通过 arguments对象访问。  
  
function sum(callback) {  
 var args = Array.prototype.slice.call(arguments, 1);  
 typeof callback === 'function' && callback(args);  
 console.log(args)  
}  
  
function add(options) {  
 var totle = 0;  
 options.map(function(item) {  
 if (typeof item === 'object' && Object.prototype.toString.call(item) === '[object Array]') {  
 item.map(function(itemCell) {  
 totle += itemCell  
 })  
 } else {  
 totle += item;  
 }  
 })  
 console.log(totle);  
}  
  
sum(add, 1, 2, 3, 4, 5, [10, 20])

回调函数是闭包的 当作为参数传递另一个函数时，回调函数将在包含函数函数体内的某个位置被执行，就像回调函数在包含函数的函数体内定义一样。这意味着回调函数是闭包的.

## 实现回调函数的基本原则

使用命名函数或匿名函数作为回调 在前面的jQuery和forEach的例子中，我们在包含函数的参数中定义匿名函数，这是使用回调函数的通用形式之一 另一个经常被使用的形式是定义一个带名称的函数，并将函数名作为参数传递给另一个函数,例如