jQuery进阶——自己开发插件

写过一些项目之后，对于js或者jq的运用也熟练了些许，慢慢的会发觉很多相似的功能在不同的项目中会运用到，每次都重新写一遍多少会有些浪费时间，又或者在特定需求下需要用到一些插件，然而经常会遇到插件并不是完全能满足需求的状况，鉴于以上种种情形，学习自己封装插件似乎势在必行。

1.在插件中使用$别名

在编写jQuery插件的时候，jQuery库必须已经加载到页面中了，为确保$别名可用，且不会发生冲突，需要用到`立即调用的函数表达式`。

```
(function($) {
    //代码部分
})(jQuery);
```

解释：这个函数接受一个参数——jQuery对象，这个参数的名字即为$。这样就可以在函数里面使用$别名了。

2.添加jQuery对象方法

实现插件功能的最好办法是扩展`jQuery.prototype对象`的方法，这样在操作DOM的时候，这些在插件中扩展的方法就可以直接拿过来使用。

```
(function($) {
    $.prototype.myMethod = function() {
        //代码部分
    }
})(jQuery);
```

2.1关键字this

在插件内部，关键字`this`引用的都是当前的jQuery对象，所以可以用this调用内置的jQuery方法，或者操作它所包含的DOM节点。

2.2隐式迭代

有一个必须要考虑到的问题，就是jQuery选择器可能会匹配多个元素，而一些方法只会检查匹配的第一个元素，所以应该独立检查和操作匹配的每一个元素，以确保匹配到的每个元素都会执行相应的方法，所以这里应该使用`.each()`方法实现隐式迭代。

```
(function($) {
    $.prototype.myMethod = function() {
        this.each(function(){
            //代码部分
        });
    }
})(jQuery);
```

2.3方法连缀

在jQuery中方法连缀是我们经常使用的，能够极大程度上的让代码简洁许多，所以，在自己开发的插件中，能够实现这样的效果也是极好的，而方法也很简单，使用`return`返回上面.each()对于this的迭代结果即可。

```
(function($) {
    $.prototype.myMethod = function() {
        return this.each(function(){
            //代码部分
        });
    }
})(jQuery);
```

3.可配置参数

3.1参数

显然插件的使用必须能满足多种场合多种需求，因此一个较完善的插件也必须是能够让使用者对参数进行自定义配置，而参数对象除了一些简单的属性，也可以是一些`回调函数`，它可以极大程度的增加插件的灵活性。

3.2默认值

同时，如果使用者需要的功能限制得并不十分严格，并不需要去自己规定参数，这种情况下，我们必须要有默认的参数值

3.3自定义默认值

还有一种情况就是，这个插件需要用到的地方非常多，每一次都要去配置参数也是一件比较麻烦的事，所以最好的办法就是可以修改默认值，这样修改一次，其他地方就可以直接使用，一劳永逸。

```
(function($) {
    $.prototype.myMethod = function(opts) {
        var options = $.extentd({},$.prototype.defaults,opts);

        return this.each(function(){
            //代码部分
        });
    };
    $.prototype.myMethod.defaults = {
        name1: value1,
        name2: value2,
        name3: value3,
        function_name1: function(){
            //回调函数
        }
    };

})(jQuery);
```

张玥

2016.6.26

书于杜比环绕无降噪装修声中