46 第三十二章 jQuery 工具函数 $.ajax（五）

jQuery [2]

`$`().xxx() 是jQuery对象方法
`$`.xxx() 工具方法，即可以给jQuery对象用，也可以给原生js对象用

一 常用工具函数

`jQuery.type()` 判断类型

`jQuery.each(` object, [callback] `)` 通用例遍方法，可用于例遍任何对象 如对象和数组。

```
1. $.each( [0,1,2], function(i, n){
2.   alert( "Item #" + i + ": " + n );
3. });
4.
5. $.each( { name: "John", lang: "JS" }, function(i, n){
6.   alert( "Name: " + i + ", Value: " + n );
7. });
```

`jQuery.proxy(` function,context `)` 改变 this指向

function:将要被改变作用域的函数
context:一个object，那个函数的作用域会被设置到这个object上来。即function的内部的this

```
    var obj = {
  name: "John",
  test: function() {
    alert( this.name );
  }
};

$("#test").click( jQuery.proxy( obj, "test" ) );

// 以下代码跟上面那句是等价的:
// $("#test").click( jQuery.proxy( obj.test, obj ) );

// 可以与单独执行下面这句做个比较。
// $("#test").click( obj.test );


```

`jQuery.noConflict(`bealoon`)` 防止冲突 可以传递布尔值，

```
$.noConflict(); //将变量$释放
console.log( $ ); //undefined
console.log(jQuery);//jQuery 依然可以使用

var a = $.noConflict();
console.log( a ); //此时a具有了$功能

```

```
var s = $.noConflict(true);
console.log(jQuery);//undefined 也将不能使用
```

二 HTTP AJAX

1、load( )

$(selector).`load(` url,data,function(response,status,xhr)`)`
load() 方法通过 AJAX 请求从服务器加载数据，并把返回的数据放置到指定的元素中。`需后台环境`，默认get请求
`url` 规定要将请求发送到哪个 URL。
`data` 可选。规定连同请求发送到服务器的数据。
`function(response,status,xhr)`
可选。规定当请求完成时运行的函数。
额外的参数：
`response` - 包含来自请求的结果数据
`status` - 包含请求的状态（”success”, “notmodified”, “error”, “timeout” 或 “parsererror”）
`xhr` - 包含 XMLHttpRequest 对象

1、加载文本

```
$('#box').click(function(){
    $('#box').load('test.txt');

  })
```

2、加载html

```
$("#result").load("ajax/test.html");
```

3、带回调函数

```
$('#box').click(function(){
        $('#box').load('test.html',function(){
                console.log('ok')
          });

})
```

5、POST形式加载

```
$('#box').click(function(){
      $('#box').load('test.php',{age:25},function(){
              console.log('ok')
       });

    })

```

4、碎片化加载，`注意` `#前面得有空格`

```
$('#box').click(function(){
            $('#box').load('test.php',{age:25},function(){
                    console.log('ok')
            });

         })

```

2、 ajax()

`jQuery.ajax()` : json形式的配置参数

`type` 请求类型

请求方式 (“POST” 或 “GET”)， 默认为 “`GET`“。

`url` 请求url

`async` 是否异步

> (默认: true) 默认设置下，所有请求均为异步请求。如果需要发送同步请求，请将此选项设置为 false。注意，同步请求将锁住浏览器，用户其它操作必须等待请求完成才可以执行。

`data` Object,String

发送到服务器的数据。将自动转换为请求字符串格式。GET 请求中将附加在 URL 后。查看 processData 选项说明以禁止此自动转换。必须为 Key/Value 格式。如果为数组，jQuery 将自动为不同值对应同一个名称。如 {foo:[“bar1”, “bar2”]} 转换为 “&foo=bar1&foo=bar2”。

`dataType` 预期服务器返回的数据类型

“ `xml` “: 返回 XML 文档，可用 jQuery 处理。
“ `html` “: 返回纯文本 HTML 信息；包含的script标签会在插入dom时执行。
“ `script` “: 返回纯文本 JavaScript 代码。不会自动缓存结果。除非设置了”cache”参数。”’注意：”’在远程请求时(不在同一个域下)，所有POST请求都将转为GET请求。(因为将使用DOM的script标签来加载)
“ `json` “: 返回 JSON 数据 。
“ `jsonp` “: JSONP 格式。使用 JSONP 形式调用函数时，如 “myurl?callback=?” jQuery 将自动替换 ? 为正确的函数名，以执行回调函数。
“ `text` “: 返回纯文本字符串

`cache` 缓存

(默认: `true` ,dataType为script和jsonp时默认为false) jQuery 1.2 新功能，设置为 false 将不缓存此页面。

`success`

success(response, Status, jqXHR)

`error`

function (xhr, status, errorMessage)
有以下三个参数：XMLHttpRequest 对象、错误信息、（可选）捕获的异常对象

`contentType`

(默认: “application/x-www-form-urlencoded”) 发送信息至服务器时内容编码类型。默认值适合大多数情况。如果你明确地传递了一个content-type给 $.ajax() 那么他必定会发送给服务器（即使没有数据要发送）

`jsonp`

在一个jsonp请求中重写回调函数的名字。这个值用来替代在”callback=?”这种GET或POST请求中URL参数里的”callback”部分，比如{jsonp:’onJsonPLoad’}会导致将”onJsonPLoad=?”传给服务器。

`jsonpCallback`

为jsonp请求指定一个回调函数名。这个值将用来取代jQuery自动生成的随机函数名。这主要用来让jQuery生成度独特的函数名，这样管理请求更容易，也能方便地提供回调函数和错误处理。你也可以在想让浏览器缓存GET请求的时候，指定这个回调函数名。

`context`

这个对象用于设置Ajax相关回调函数的上下文。也就是说，让回调函数内this指向这个对象（如果不设定这个参数，那么this就指向调用本次AJAX请求时传递的options参数）。比如指定一个DOM元素作为context参数，这样就设置了success回调函数的上下文为这个DOM元素。就像这样：

```
1. $.ajax({ url: "test.html", context: document.body, success: funct
   ion(){
2.     $(this).addClass("done");
3. }});
```

3、`jQuery.get( )` 方法通过远程 HTTP GET 请求载入信息。

这是一个简单的 GET 请求功能以取代复杂
.ajax。请求成功时可调用回调函数。如果需要在出错时执行函数，请使用.ajax。

> 语法
> `$(selector).get(url,data,success(response,status,xhr),dataType)`

url 必需。规定将请求发送的哪个 URL。
data 可选。规定连同请求发送到服务器的数据。
success(response,status,xhr) 可选。规定当请求成功时运行的函数。

> - response - 包含来自请求的结果数据
> - status - 包含请求的状态
> - xhr - 包含 XMLHttpRequest 对象

dataType 可选。规定预计的服务器响应的数据类型。默认地，jQuery 将智能判断。
可能的类型：”xml”“html”“text”“script”“json”“jsonp”

等价于

```
$.ajax({
  url: url,
  data: data,
  success: success,
  dataType: dataType
});
```

```
$.get("test.php");
$.get("test.php", { name: "John", time: "2pm" } );
$.get("test.php", function(data){
  alert("Data Loaded: " + data);
});

$.get("test.cgi", { name: "John", time: "2pm" },
  function(data){
    alert("Data Loaded: " + data);
  });
```

4、`jQuery.post( )`post() 方法通过 HTTP POST 请求从服务器载入数据。

> 语法
> `jQuery.post(url,data,success(data, textStatus, jqXHR),dataType)`

`url`必需。规定把请求发送到哪个 URL。
`data` 可选。映射或字符串值。规定连同请求发送到服务器的数据。
`success(data, textStatus, jqXHR)` 可选。请求成功时执行的回调函数。
`dataType`可选。规定预期的服务器响应的数据类型。
默认执行智能判断（xml、json、script 或 html）。

> 等价于

```
$.ajax({
  type: 'POST',
  url: url,
  data: data,
  success: success,
  dataType: dataType
});
```

```
$.post('05-2 $.ajax.php',{age:25},function(res,sta,xhr){

    console.log(res,sta,xhr)

},'json')
```

5、`jQuery.getJSON( )` 通过 HTTP GET 请求载入 JSON 数据。

> 语法
> `jQuery.getJSON(url,data,success(data,status,xhr))`

`url` 必需。规定将请求发送的哪个 URL。
`data` 可选。规定连同请求发送到服务器的数据。
`success(data,status,xhr)` 可选。规定当请求成功时运行的函数。
response - 包含来自请求的结果数据
status - 包含请求的状态
xhr - 包含 XMLHttpRequest 对象

```
$.post('05-3 $.ajax.php',{age:25},function(res,sta,xhr){
                        console.log(res,sta,xhr)
                },'json')
```

三 继承

`jQuery.extend(` [deep], target, object1, [objectN] `)`
用一个或多个其他对象来扩展一个对象，返回被扩展的对象。

> 如果不指定target，则给jQuery命名空间本身进行扩展。这有助于插件作者为jQuery增加新方法。 如果第一个参数设置为true，则jQuery返回一个深层次的副本，递归地复制找到的任何对象。否则的话，副本会与原对象共享结构。 未定义的属性将不会被复制，然而从对象的原型继承的属性将会被复制。
> `deep` :如果设为true，则递归合并。 如果第一个参数设置为true，则jQuery返回一个深层次的副本，递归地复制找到的任何对象。
> `target` :待修改对象。一个对象，如果附加的对象被传递给这个方法将那么它将接收新的属性，如果它是唯一的参数将扩展jQuery的命名空间。
> `object1` :待合并到第一个对象的对象。
> `objectN` :待合并到第一个对象的对象。

```
var empty = {};
var defaults = { validate: false, limit: 5, name: "foo" };
var options = { validate: true, name: "bar" };
var settings = jQuery.extend(empty, defaults, options);
```

> 结果：

```
settings == { validate: true, limit: 5, name: "bar" }
empty == { validate: true, limit: 5, name: "bar" }
```

四插件拓展

> jQuery为开发插件提拱了两个方法.

`jQuery.extend` 添加工具函数，给构造函数添加方法

```
var obj = { age:function(){alert(888);}  };
 $.extend(obj);
 $.age();//888

```

等价于

```
$.age = function(){alert(888);}
 $.age();//888
```

`jQuery.fn.extend` 给对象添加方法

```
var obj = {
    run:function(){
          alert('run');
      }
  }
 $.fn.extend(obj);
 $('#box').run();//run
```

```
1.      console.log( $.fn === $.prototype ) //true
```