**jquery中的工具、Callbacks以及Deferred进行简单讲解**

## **jQuery.support**

### V1.3概述

jQuery 1.3 新增。一组用于展示不同浏览器各自特性和bug的属性集合。

jQuery提供了一系列属性，你也可以自由增加你自己的属性。其中许多属性是很低级的，所以很难说他们能否在日新月异的发展中一直保持有效，但这这些主要用于插件和内核开发者。

所有这些支持的属性值都通过特性检测来实现，而不是用任何浏览器检测。以下有一些非常棒的资源用于解释这些特性检测是如何工作的：

* http://peter.michaux.ca/articles/feature-detection-state-of-the-art-browser-scripting
* http://yura.thinkweb2.com/cft/
* http://www.jibbering.com/faq/faq\_notes/not\_browser\_detect.html

jQuery.support主要包括以下测试：

****boxModel****: 如果这个页面和浏览器是以W3C CSS盒式模型来渲染的，则等于true。通常在IE 6和IE 7的怪癖模式中这个值是false。在document准备就绪前，这个值是null。

****cssFloat****: 如果用cssFloat来访问CSS的float的值，则返回true。目前在IE中会返回false,他用styleFloat代替。

****hrefNormalized****: 如果浏览器从getAttribute("href")返回的是原封不动的结果，则返回true。在IE中会返回false，因为他的URLs已经常规化了。

****htmlSerialize****: 如果浏览器通过innerHTML插入链接元素的时候会序列化这些链接，则返回true，目前IE中返回false。

****leadingWhitespace****: 如果在使用innerHTML的时候浏览器会保持前导空白字符，则返回true，目前在IE 6-8中返回false。

****noCloneEvent****: 如果浏览器在克隆元素的时候不会连同事件处理函数一起复制，则返回true，目前在IE中返回false。

****objectAll****: 如果在某个元素对象上执行getElementsByTagName("\*")会返回所有子孙元素，则为true，目前在IE 7中为false。

****opacity****: 如果浏览器能适当解释透明度样式属性，则返回true，目前在IE中返回false，因为他用alpha滤镜代替。

****scriptEval****: 使用 appendChild/createTextNode 方法插入脚本代码时，浏览器是否执行脚本，目前在IE中返回false，IE使用 .text 方法插入脚本代码以执行。

****style****: 如果getAttribute("style")返回元素的行内样式，则为true。目前IE中为false，因为他用cssText代替。

****tbody****: 如果浏览器允许table元素不包含tbody元素，则返回true。目前在IE中会返回false，他会自动插入缺失的tbody。

### 示例

#### 描述:

检测浏览器是否支持盒式模型

##### jQuery 代码:

jQuery.support.boxModel

## **jQuery.browser.version**

### V1.1.3概述

在jQuery 1.3中不建议使用。浏览器渲染引擎版本号。

典型结果: Internet Explorer: 6.0, 7.0

Mozilla/Firefox/Flock/Camino: 1.7.12, 1.8.1.3

Opera: 9.20

Safari/Webkit: 312.8, 418.9

### 示例

#### 描述:

显示当前 IE 浏览器版本号。

##### jQuery 代码:

if ( $.browser.msie )

alert( $.browser.version );

## **jQuery.boxModel**

### V1.0概述

在jQuery 1.3中不建议使用。当前页面中浏览器是否使用标准盒模型渲染页面。 建议使用 jQuery.support.boxModel 代替。W3C CSS 盒模型.

### 示例

#### 描述:

在 Internet Explorer 怪癖模式（QuirksMode）中返回 False。

##### jQuery 代码:

$.boxModel

## **jQuery.each(object, [callback])**

### 概述

通用例遍方法，可用于例遍对象和数组。

不同于例遍 jQuery 对象的 $().each() 方法，此方法可用于例遍任何对象。回调函数拥有两个参数：第一个为对象的成员或数组的索引，第二个为对应变量或内容。如果需要退出 each 循环可使回调函数返回 false，其它返回值将被忽略。

### 参数

#### ****object,[callback]****Object,FunctionV1.0

****object****:需要例遍的对象或数组。

****callback****:每个成员/元素执行的回调函数。

### 示例

#### 描述:

例遍数组，同时使用元素索引和内容。

##### jQuery 代码:

$.each( [0,1,2], function(i, n){

alert( "Item #" + i + ": " + n );

});

#### 描述:

例遍对象，同时使用成员名称和变量内容。

##### jQuery 代码:

$.each( { name: "John", lang: "JS" }, function(i, n){

alert( "Name: " + i + ", Value: " + n );

});

## **jQuery.extend([deep], target, object1, [objectN])**

### 概述

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。

### 参数

#### ****target,[object1],[objectN]****Object,Object,ObjectV1.0

****target****: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。

#### ****[deep],target,object1,[objectN]****Object,Object,Object,ObjectV1.1.4

****deep****:如果设为true，则递归合并。

****target****:待修改对象。

****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。

### 示例

#### 描述:

合并 settings 和 options，修改并返回 settings。

##### jQuery 代码:

var settings = { validate: false, limit: 5, name: "foo" };

var options = { validate: true, name: "bar" };

jQuery.extend(settings, options);

##### 结果:

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

#### 描述:

合并 defaults 和 options, 不修改 defaults。

##### jQuery 代码:

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" }

## **jQuery.grep(array, callback, [invert])**

### 概述

使用过滤函数过滤数组元素。

此函数至少传递两个参数：待过滤数组和过滤函数。过滤函数必须返回 true 以保留元素或 false 以删除元素。

### 参数

#### ****array,callback,[invert]****Array,Function,BooleanV1.0

****array****:待过滤数组。

****callback****:此函数将处理数组每个元素。第一个参数为当前元素，第二个参数而元素索引值。此函数应返回一个布尔值。另外，此函数可设置为一个字符串，当设置为字符串时，将视为“lambda-form”（缩写形式？），其中 a 代表数组元素，i 代表元素索引值。如“a > 0”代表“function(a){ return a > 0; }”。

****invert****:如果 "invert" 为 false 或为设置，则函数返回数组中由过滤函数返回 true 的元素，当"invert" 为 true，则返回过滤函数中返回 false 的元素集。

### 示例

#### 描述:

过滤数组中小于 0 的元素。

##### jQuery 代码:

$.grep( [0,1,2], function(n,i){

return n > 0;

});

##### 结果:

[1, 2]

#### 描述:

排除数组中大于 0 的元素，使用第三个参数进行排除。

##### jQuery 代码:

$.grep( [0,1,2], function(n,i){

return n > 0;

}, true);

##### 结果:

[0]

## **jQuery.sub()**

### V1.5概述

可创建一个新的jQuery副本，不影响原有的jQuery对像。

有两个具体使用jQuery.sub（）创建案例。首先是提供完全没有破坏jQuery原有一切的方法，另一个用于帮助做jQuery插件封装和基本命名空间。

请注意，jQuery.sub（）不会做任何特殊的隔离 - 这不是它的意图。所有关于jQuery的sub'd版本的方法将仍然指向原来的jQuery。（绑定和触发仍将通过主jQuery的事件，数据将通过主绑定的元素的jQuery，Ajax的查询和活动将通过主jQuery的运行，等等）。

请注意，如果你正在寻找使用这个开发插件，应首先认真考虑使用一些类似jQuery UI widget工厂，这两个状态和插件管理子方法。 [使用jQuery UI widget的一些例子](http://blog.nemikor.com/2010/05/15/building-stateful-jquery-plugins/)建立一个插件。

这种方法的具体使用情况下可以通过一些例子最好的描述。

### 参数

#### ****jQuery.sub()****

### 示例

#### 描述:

添加一个jQuery的方法，以便它不会受到外部分：

##### jQuery 代码:

(function(){

var sub$ = jQuery.sub();

sub$.fn.myCustomMethod = function(){

return 'just for me';

};

sub$(document).ready(function() {

sub$('body').myCustomMethod() // 'just for me'

});

})();

typeof jQuery('body').myCustomMethod // undefined

#### 描述:

改写一些jQuery的方法，以提供新的功能。

##### jQuery 代码:

(function() {

var myjQuery = jQuery.sub();

myjQuery.fn.remove = function() {

// New functionality: Trigger a remove event

this.trigger("remove");

// Be sure to call the original jQuery remove method

return jQuery.fn.remove.apply( this, arguments );

};

myjQuery(function($) {

$(".menu").click(function() {

$(this).find(".submenu").remove();

});

// A new remove event is now triggered from this copy of jQuery

$(document).bind("remove", function(e) {

$(e.target).parent().hide();

});

});

})();

// Regular jQuery doesn't trigger a remove event when removing an element

// This functionality is only contained within the modified 'myjQuery'.

#### 描述:

创建一个插件，它返回插件的具体办法。

##### jQuery 代码:

(function() {

// Create a new copy of jQuery using sub()

var plugin = jQuery.sub();

// Extend that copy with the new plugin methods

plugin.fn.extend({

open: function() {

return this.show();

},

close: function() {

return this.hide();

}

});

// Add our plugin to the original jQuery

jQuery.fn.myplugin = function() {

this.addClass("plugin");

// Make sure our plugin returns our special plugin version of jQuery

return plugin( this );

};

})();

$(document).ready(function() {

// Call the plugin, open method now exists

$('#main').myplugin().open();

// Note: Calling just $("#main").open() won't work as open doesn't exist!

});

## **jQuery.when(deferreds)**

### 概述

提供一种方法来执行一个或多个对象的回调函数，延迟对象通常表示异步事件。

如果单一延迟传递给jQuery.when ，它是通过这个方法和延迟对象附加的其他方法可访问绑定的回调函数返回的，如[defered.then](http://api.jquery.com/deferred.then/) 。当延迟得到解决或者拒绝，通常的代码创建了原来的延迟，适当的回调将被调用。例如，jqXHR对象返回jQuery.ajax是一个延期，可以用这种方式：

$.when( $.ajax("test.aspx") ).then(function(ajaxArgs){

alert(ajaxArgs[1]); /\* ajaxArgs is [ "success", statusText, jqXHR ] \*/

});

如果一个参数被传递给jQuery.when ，这不是延迟，这将被视为延迟解决，并立即将执行附加任何doneCallbacks。该doneCallbacks传递原始的参数。在这种情况下，任何failCallbacks您可能会设置是永远不会被调用，因为延迟从不拒绝。

例如：

$.when( { testing: 123 } ).done(

function(x){ alert(x.testing); } /\* alerts "123" \*/

);

在案例中有多个延迟对象传递jQuery.when ，该方法返回一个新的“宿主”延迟对象，跟踪所有已通过Deferreds聚集状态。该方法能够解决它的“宿主”延迟尽快解决所有延迟，或拒绝尽快将被拒绝的延迟。如果“宿主”延迟得到解决，它是通过传递给解析值，所有的延迟 jQuery.when 。例如，当延迟是jQuery.ajax()请求，参数将是jqXHR对象的要求，以便他们在名单内的说法。

在多延迟情况下，如果延迟一被拒绝，jQuery.when火灾立即掌握其推迟failCallbacks。请注意，延迟一些可能仍然在这一点没有得到解决。如果您需要执行额外的处理对于这种情况，如取消任何未完成的Ajax请求，你可以保持基本jqXHR引用对象在封闭和检查/取消在failCallback他们。

### 参数

#### ****deferreds****V1.5

一个或多个延迟对象，或者普通的JavaScript对象。

### 示例

#### 描述:

执行Ajax请求后两个函数是成功的。（见jQuery.ajax()对于一个成功的和错误的案件为AJAX请求的完整描述文档）。

##### jQuery 代码:

$.when($.ajax("/page1.php"), $.ajax("/page2.php")).done(function(a1, a2){

/\* a1 and a2 are arguments resolved for the

page1 and page2 ajax requests, respectively \*/

var jqXHR = a1[2]; /\* arguments are [ "success", statusText, jqXHR ] \*/

if ( /Whip It/.test(jqXHR.responseText) ) {

alert("First page has 'Whip It' somewhere.");

}

});

执行函数myfunc当两个Ajax请求是成功的，如果任一或myFailure有一个错误。

##### jQuery 代码:

$.when($.ajax("/page1.php"), $.ajax("/page2.php"))

.then(myFunc, myFailure);

## **jQuery.makeArray(obj)**

### 概述

将类数组对象转换为数组对象。

类数组对象有 length 属性，其成员索引为 0 至 length - 1。实际中此函数在 jQuery 中将自动使用而无需特意转换。

### 参数

#### ****obj****ObjectV1.2

类数组对象。

### 示例

#### 描述:

过滤数组中小于 0 的元素。

##### HTML 代码:

<div>First</div><div>Second</div><div>Third</div><div>Fourth</div>

##### jQuery 代码:

var arr = jQuery.makeArray(document.getElementsByTagName("div"));

arr.reverse(); // 使用数组翻转函数

##### 结果:

Fourth

Third

Second

First

## **jQuery.map(arr|obj,callback)**

### 概述

将一个数组中的元素转换到另一个数组中。

作为参数的转换函数会为每个数组元素调用，而且会给这个转换函数传递一个表示被转换的元素作为参数。转换函数可以返回转换后的值、null（删除数组中的项目）或一个包含值的数组，并扩展至原始数组中。

### 参数

#### ****array,callback****Array,FunctionV1.0

****array****:待转换数组。

****callback****Array:为每个数组元素调用，而且会给这个转换函数传递一个表示被转换的元素作为参数。函数可返回任何值。另外，此函数可设置为一个字符串，当设置为字符串时，将视为“lambda-form”（缩写形式？），其中 a 代表数组元素。如“a \* a”代表“function(a){ return a \* a; }”。

#### ****arrayOrObject,callback****Array/Object,FunctionV1.6

****arrayOrObject****:数组或者对象。

为每个数组元素调用，而且会给这个转换函数传递一个表示被转换的元素作为参数。函数可返回任何值。另外，此函数可设置为一个字符串，当设置为字符串时，将视为“lambda-form”（缩写形式？），其中 a 代表数组元素。如“a \* a”代表“function(a){ return a \* a; }”。

### 示例

#### 描述:

将原数组中每个元素加 4 转换为一个新数组。

##### jQuery 代码:

$.map( [0,1,2], function(n){

return n + 4;

});

##### 结果:

[4, 5, 6]

#### 描述:

原数组中大于 0 的元素加 1 ，否则删除。

##### jQuery 代码:

$.map( [0,1,2], function(n){

return n > 0 ? n + 1 : null;

});

##### 结果:

[2, 3]

#### 描述:

原数组中每个元素扩展为一个包含其本身和其值加 1 的数组，并转换为一个新数组。

##### jQuery 代码:

$.map( [0,1,2], function(n){

return [ n, n + 1 ];

});

##### 结果:

[0, 1, 1, 2, 2, 3]

## **jQuery.inArray(value,array,[fromIndex])**

### 概述

确定第一个参数在数组中的位置，从0开始计数(如果没有找到则返回 -1 )。

### 参数

#### ****value,array,[fromIndex]****Any,Array,NumberV1.2

****value****:用于在数组中查找是否存在

****array****:待处理数组。

****fromIndex****:用来搜索数组队列，默认值为0。

#### ****array****Array

待处理数组。

### 示例

#### 描述:

查看对应元素的位置

##### jQuery 代码:

var arr = [ 4, "Pete", 8, "John" ];

jQuery.inArray("John", arr); //3

jQuery.inArray(4, arr); //0

jQuery.inArray("David", arr); //-1

jQuery.inArray("Pete", arr, 2); //-1

## **jQuery.merge(first,second)**

### 概述

合并两个数组

返回的结果会修改第一个数组的内容——第一个数组的元素后面跟着第二个数组的元素。要去除重复项，请使用$.unique()

### 参数

#### ****first,second****Array,ArrayV1.0

****first****:第一个待处理数组，会改变其中的元素。

****second****:第二个待处理数组，不会改变其中的元素。

### 示例

#### 描述:

合并两个数组到第一个数组上。

##### jQuery 代码:

$.merge( [0,1,2], [2,3,4] )

##### 结果:

[0,1,2,2,3,4]

## **jQuery.unique(array)**

### 概述

删除数组中重复元素。只处理删除DOM元素数组，而不能处理字符串或者数字数组。

### 参数

#### ****array****ArrayV1.1.3

待处理数组。

### 示例

#### 描述:

删除重复 div 标签。

##### jQuery 代码:

$.unique(document.getElementsByTagName("div"));

##### 结果:

[<div>, <div>, ...]

## **jQuery.parseJSON(json)**

### 概述

接受一个JSON字符串，返回解析后的对象。

传入一个畸形的JSON字符串会抛出一个异常。比如下面的都是畸形的JSON字符串：

* {test: 1} （ test 没有包围双引号）
* {'test': 1} （使用了单引号而不是双引号）

另外，如果你什么都不传入，或者一个空字符串、null或undefined，parseJSON都会返回 null 。

### 参数

#### ****json****StringV1.4.1

要解析的JSON字符串

### 示例

#### 描述:

解析一个JSON字符串

##### jQuery 代码:

var obj = jQuery.parseJSON('{"name":"John"}');

alert( obj.name === "John" );