jQuery（3）---源码

2014年8月27日 10:23

http://www.cnblogs.com/chyingp/archive/2013/06/03/jquery-souce-code-study.html
http://www.cnblogs.com/nuysoft/archive/2011/11/14/2248023.html

jquery2.0和1.0的区别是：2.0将不支持IE6/7/8。
http://www.cnblogs.com/nuysoft/
http://nuysoft.iteye.com/
jquery源码阅读，这位作者出了一本书叫jquery技术内幕
http://item.jd.com/1080321026.html

什么是JQ？
一个优秀的JS库，大型开发必备
JQ好处？
简化JS的复杂操作
不需要关心兼容性
提供大量实用方法

最前面的（87）代表源码中的行数
==========
miaov1~3
版本2.0.3
结构：
(function(){
（21,94） 定义了一些变量和函数，重要函数：jQuery = function(selector, context){}
（96, 280）给jq对象添加方法和属性
（285, 347） extend：JQ的继承方法
（349, 817） jQuery.extend()：扩展工具方法，比如$.trim();工具方法既可以给jquery对象用，也可以给原生js用
(877, 2856） sizzle 复杂元素选择器的实现
（2880, 3042） Callbacks：回调对象：作用是函数的统一管理
（3043, 3183） Deffered：延迟对象：作用是对异步的统一管理
（3184, 3295） support：功能检测
（3308, 3652） data()：数据缓存，避免大数据添加到元素身上 造成内存泄露
（3653, 3797） queue()：队列管理
（3803, 4299） attr() prop() val() addClass()等，对元素属性的操作
（4300, 5128） on() trigger()：事件操作的相关方法
(5140, 6057) DOM操作方法：添加 删除 获取 包装 DOM筛选
(6058, 6620) css()：样式的操作
(6621, 7854) 提交的数据和ajax()：ajax() load() getJson()
(7855, 8584) animate()： 运动的方法
(8585, 8792) offset()：位置和尺寸的方法
(8804, 8821) JQ支持模块化的模式
(8826) window.jQuery = window.$ = jQuery; 通过外面找到jQuery，对外提供的接口
})();

（21,94） ==========
（21,94） 定义了一些变量和函数，重要函数：jQuery = function(selector, context){} miaov4~6，

	/*! * jQuery JavaScript Library v2.0.3 * http: //jquery.com/ * * Includes Sizzle.js * http: //sizzlejs.com/ * * Copyright 2005, 2013 jQuery Foundation, Inc. and other contributors * Released under the MIT license * http: //jquery.org/license * * Date: 2013-07-03T13:30Z */
（14）为什么使用window？ (function(){ window })(window); 1.使用局部变量查找速度快 2.传参可以进行压缩 为什么传参undefined？ 防止undefined被修改	(function(window, undefined) {
（20）"use strict"：不建议去掉注释，去掉会造成假死状态等。目前此bug已修复	// Can't do this because several apps including ASP.NET trace // the stack via arguments.caller.callee and Firefox dies if // you try to trace through "use strict" call chains. (#13335) // Support: Firefox 18+ //"use strict";
（23）rootjQuery： 1.方便压缩；2.定义一个变量方便后期可维护	var // A central reference to the root jQuery(document) rootjQuery,
（26）readyList： 跟DOM加载有关	// The deferred used on DOM ready readyList,
（30）core_strundefined：得到的是字符串形式的undefined window.a == undefined; typeof window.a == 'undefined'; 一般情况下，以上两种方式判断是ok的。 但是在老版本浏览器下IE9，当判断的是一个xmlNode时，判断不出来。所以最好采用typeof的方法进行判断。	// Support: IE9 // For `typeof xmlNode.method` instead of `xmlNode.method !== undefined` core_strundefined = typeof undefined,
（33）对window下的一些变量进行存储，方便压缩	// Use the correct document accordingly with window argument (sandbox) location = window.location,

	document = window.document, docElem = document.documentElement,
（38，41）防冲突，如果在引入jquery库之前已经定义了jQuery或者$就先暂存起来	// Map over jQuery in case of overwrite _jQuery = window.jQuery, // Map over the $ in case of overwrite _$ = window.$,
（44）使用$.type()时用到的，它里面可能会存成这样的 class2type = { '[Object String]' : 'string' }	// [[Class]] -> type pairs class2type = {},
（47）没有什么实际用处了，老版本中是和数据缓存有关的	// List of deleted data cache ids, so we can reuse them core_deletedIds = [],
（49）版本号	core_version = "2.0.3",
（52,58）存一些方法名	// Save a reference to some core methods core_concat = core_deletedIds.concat, core_push = core_deletedIds.push, core_slice = core_deletedIds.slice, core_indexOf = core_deletedIds.indexOf, core_toString = class2type.toString, core_hasOwn = class2type.hasOwnProperty, core_trim = core_version.trim,
（61）jQuery方法，fn就是prototype， jQuery.fn.init是一个构造函数 function Aaa(){} Aaa.prototype.init = function(){}; Aaa.prototype.css = function(){}; var a1 = new Aaa(); a1.init(); a1.css(); 下面的与上面的不同，可以一目了然的找到css，不用做烦琐的初始化工作 function jQuery(){ return new jQuery.prototype.init(); } jQuery.prototype.init = function(){}; jQuery.prototype.css = function(){}; jQuery.prototype.init.prototype = jQuery.prototype; jQuery().css(); 283行有：jQuery.fn.init.prototype = jQuery.prototype; 故在 jQuery.prototype下添加的任何方法，可以通过jQuery.prototype.init new出来的对象使用	// Define a local copy of jQuery jQuery = function(selector, context) { // The jQuery object is actually just the init constructor 'enhanced' return new jQuery.fn.init(selector, context, rootjQuery); },
（67）一些正则，在css方法时使用 core_pnum 匹配数字 core_rnotwhite 匹配单词 rquickExpr 匹配标签和id，通过location.hash防止XSS注入 aaa 或 #id rsingleTag 匹配独立的空标签 rmsPrefix 匹配前缀 margin-left ： marginLeft -webkit-margin-left : webkitMarginLeft -ms-margin-left : MsMarginLeft 和其他的不一样，M要大写 rdashAlpha 找到-和字符，转大小写 -left : Left -2d : 2d	// Used for matching numbers core_pnum = /[+-]?(?:\d*\.|)\d+(?:[eE][+-]?\d+|)/.source, // Used for splitting on whitespace core_rnotwhite = /\S+/g, // A simple way to check for HTML strings // Prioritize #id over <tag> to avoid XSS via location.hash (#9521) // Strict HTML recognition (#11290: must start with <) rquickExpr = /^(?:\s*(<[\w\W]+>)[^>]*|#([\w-]*))$/, // Match a standalone tag rsingleTag = /^<(\w+)\s*\/?>(?:<\/\1>\|)$/, // Matches dashed string for camelizing rmsPrefix = /^-ms-/, rdashAlpha = /-([\da-z])/gi,
（85）转驼峰的回调函数	// Used by jQuery.camelCase as callback to replace() fcamelCase = function(all, letter) { return letter.toUpperCase(); },
（90）DOM加载成功之后会触发的。819行相关。 先取消两个事件的绑定，然后调用工具方法jQquery.ready()。 最终只会走一次jQuery.ready()，这个ready相关代码在382行。	// The ready event handler and self cleanup method completed = function() { document.removeEventListener("DOMContentLoaded", completed, false); window.removeEventListener("load", completed, false); jQuery.ready(); };

（96, 280）==========

（285, 347）==========

（349, 846）==========

（877, 2856 ）==========

（2880, 3042）==========

（3043, 3183）==========

（3184, 3295）==========

（3184, 3295） support：功能检测 miaov 39~42

support在这个版本中做的处理不多。多的是在1.11等版本中，做IE678的兼容处理。

```
for(var attr in $.support){
    $.('body').append( '<div>' + attr + ' : ' + $.support[attr] + '</div>');
}
```

然后呢，会得到这么个结果（不同浏览器的truefalse不一样）

CheckOn : true

optSelected : false

reliableMarginRight : true

boxSizingReliable : false

pixelPosition : true

noCloneChecked : true

optDisabled : true

radioValue : false
checkClone : true
focusinBubbles : true
clearCloneStyle : false
cors : true
ajax : true
boxSizing : true

support只是用于检测，hooks来解决兼容问题

（3184）创建一些元素，通过元素的兼容表现来看是否支持一些功能。

```
jQuery.support = (function( support ) {
    var input = document.createElement("input"),
        fragment = document.createDocumentFragment(),
        div = document.createElement("div"),
        select = document.createElement("select"),
        opt = select.appendChild( document.createElement("option") );
```

（3192）这个没太大必要，最新版本中已经去掉了这个if判断，因为input默认会是text，所有浏览器都会走后面。

```
    // Finish early in limited environments
    if ( !input.type ) {
        return support;
    }
```

（3196）将input改成了复选框

```
    input.type = "checkbox";
```

（3200）看复选框的默认value值到底是什么，默认大部分情况下是”on“，但是在老版本的webkit下是空的。那么，在老版本下就要做处理让它默认是on了，这个处理的代码在4292行

```
    // Support: Safari 5.1, iOS 5.1, Android 4.x, Android 2.3
    // Check the default checkbox/radio value ("" on old WebKit; "on" elsewhere)
    support.checkOn = input.value !== "";
```

（3204）opt.selected是下拉菜单的子项，返回true，说明被选中了，返回false，说明没有被选中。

火狐，chrome下，创建一个下拉菜单，默认情况下，第一个子项是被选中的，但是在IE下是不选中的。

```
    // Must access the parent to make an option select properly
    // Support: IE9, IE10
    support.optSelected = opt.selected;
```

（3207）定义了初始值

有些是页面加载完就可以判断的，有些需要做dom操作再进行判断。

```
    // Will be defined later
    support.reliableMarginRight = true;
    support.boxSizingReliable = true;
    support.pixelPosition = false;
```

（3213）

让复选框选中，然后克隆一份，看是否被选中，IE9，10都是false，其他是true，让所有浏览器下复制出来的checkbox都能是选中的。

```
    // Make sure checked status is properly cloned
    // Support: IE9, IE10
    input.checked = true;
    support.noCloneChecked = input.cloneNode( true ).checked;
```

（3218）现在好像都是true，只有在老版本的webkit下才会有影响。

```
    // Make sure that the options inside disabled selects aren't marked as disabled
    // (WebKit marks them as disabled)
    select.disabled = true;
    support.optDisabled = !opt.disabled;
```

（3223）重新创建了一个input，这个一定要先设置value，再设置type。IE9，10，11都是false。其他会返回on。

```
    // Check if an input maintains its value after becoming a radio
    // Support: IE9, IE10
    input = document.createElement("input");
    input.value = "t";
    input.type = "radio";
    support.radioValue = input.value === "t";
```

（3229）判断clone出来的节点是否有属性

IE 火狐 Chrome都是true，Safari 是false，因为Safari的webkit版本低

```
    // #11217 - WebKit loses check when the name is after the checked attribute
    input.setAttribute( "checked", "t" );
    input.setAttribute( "name", "t" );

    fragment.appendChild( input );

    // Support: Safari 5.1, Android 4.x, Android 2.3
    // old WebKit doesn't clone checked state correctly in fragments
    support.checkClone = fragment.cloneNode( true ).cloneNode( true ).lastChild.checked;
```

（3240）onfocusin在IE下支持，只有FF,Chrome和Safari不支持，这个事件能冒泡，onfocus不能冒泡

```
    // Support: Firefox, Chrome, Safari
    // Beware of CSP restrictions (https: //developer.mozilla.org/en/Security/CSP)
    support.focusinBubbles = "onfocusin" in window;
```

（3242）克隆出来的div的修改不应该影响到原div，但是如果影响到了，这里就会返回false。在IE下都返回false，其他返回true。

```
    div.style.backgroundClip = "content-box";
    div.cloneNode( true ).style.backgroundClip = "";
    support.clearCloneStyle = div.style.backgroundClip === "content-box";
```

（3247）创建一些DOM节点，然后再进行判断，box-sizing 设置标准模式或者是怪异模式。

divReset 设置标准样式。

找到body标签，判断其是否存在，如果不存在，就直接return了，

```
    // Run tests that need a body at doc ready
    jQuery(function() {
        var container, marginDiv,
            // Support: Firefox, Android 2.3 (Prefixed box-sizing versions).
            divReset = "padding:0;margin:0;border:0;display:block;-webkit-box-sizing:content-box;-moz-box-
            sizing:content-box;box-sizing:content-box",
            body = document.getElementsByTagName("body")[ 0 ];
        if ( !body ) {
            // Return for frameset docs that don't have a body
            return;
        }
```

（3258）设置位置到-9999，让它到可视页面外面，margin-top是针对1.几的版本。

div.innerHTML 也是针对1.几的版本，没有整理干净。

给div设置样式，改成怪异模式，margin-top同样是1.几的版本的。

```
        container = document.createElement("div");
        container.style.cssText = "border:0;width:0;height:0;position:absolute;top:0;left:-9999px;margin-top:1px";

        // Check box-sizing and margin behavior.
        body.appendChild( container ).appendChild( div );
        div.innerHTML = "";
        // Support: Firefox, Android 2.3 (Prefixed box-sizing versions).
        div.style.cssText = "-webkit-box-sizing:border-box;-moz-box-sizing:border-box;box-sizing:border-
        box;padding:1px;border:1px;display:block;width:4px;margin-top:1%;position:absolute;top:1%";
```

（3269）swap是css转换的方法，zoom主要是可以设置页面的显示比例的。如果它不为空且不为1的时候，会影响到offsetWidth的值，所以这里要统一一下，现在所有浏览器都是true

```
        // Workaround failing boxSizing test due to offsetWidth returning wrong value
        // with some non-1 values of body zoom, ticket #13543
        jQuery.swap( body, body.style.zoom != null ? { zoom: 1 } : {}, function() {
            support.boxSizing = div.offsetWidth === 4;
        });
```

（3274）在node.js下是没有这个属性的，就不会走if。

为什么要在nodejs下进行判断呢？

里面是像素的position的判断，首先获取到top值，FFChrome IE是true，Safari是false

那就是说，如果你在一个样式的值上设置了百分比，那除了Safari以

```
        // Use window.getComputedStyle because jsdom on node.js will break without it.
        if ( window.getComputedStyle ) {
            support.pixelPosition = ( window.getComputedStyle( div, null ) || {} ).top !== "1%";
            support.boxSizingReliable = ( window.getComputedStyle( div, null ) || { width: "4px" } ).width === "4px";

            // Support: Android 2.3
```

外，其他浏览器都会把这个百分比自动的转换为像素。

boxSizingReliable ：IE下怪异模式，并还有padding时，将得到width = 2px。

创建一个div，判断marginright reliableMarginRight 都是true。

然后把创建的元素删除掉。

---

cors ajax

```
                // Support: Android 2.3
                // Check if div with explicit width and no margin-right incorrectly
                // gets computed margin-right based on width of container. (#3333)
                // WebKit Bug 13343 - getComputedStyle returns wrong value for margin-right
                marginDiv = div.appendChild( document.createElement("div") );
                marginDiv.style.cssText = div.style.cssText = divReset;
                marginDiv.style.marginRight = marginDiv.style.width = "0";
                div.style.width = "1px";

                support.reliableMarginRight =
                    !parseFloat( ( window.getComputedStyle( marginDiv, null ) || {} ).marginRight );
            }

            body.removeChild( container );
        });

        return support;
})( {} );
```

(3308, 3652) ==========

(3653, 3797) ==========

(3803, 4299) ==========

(4300, 5128) ==========

(4300, 5128) on() trigger()：事件操作的相关方法，miaov59~

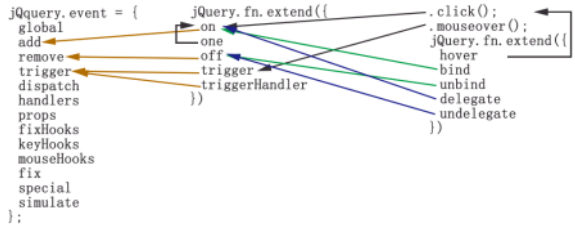
代码框架简化：

jQquery.event = {
global
add 绑定事件
remove 取消事件
trigger 主动触发事件
dispatch
handlers
props
fixHooks
keyHooks
mouseHooks
fix
special
simulate
};
jQuery.Event = function(){};
jQuery.Event.prototype = {
isDefaultPrevented
isPropagationStopped
isImmediatePropagationStopped
preventDefault
stopPropagation
stopImmediatePropagation
};
jQuery.fn.extend({
on
one
off
trigger
triggerHandler
})

(6720-6751)

.click();
.mouseover();
jQuery.fn.extend({
hover
bind
unbind
delegate
undelegate
});

大体上的调用关系：



.click mouseover调用的是one或者trigger，hover调用的moueover，bind调用的on...

看代码的顺序：

on =>

用法：

```
$('#div1').on('click', function(){
    alert(123);
});
```

```
$('#div1').on('click', {name:'hello'}, function(ev){
    alert(ev.data.name);
});
```

```
$('ul').delegate('li', 'click', function(){ //click操作是在ul身上，但是通过li是可以找到这个元素的，移动端很重要
    $(this).css('background', 'red');
});
on调用四个参数的时候和delegate是一样的，知识顺序有个微调
$('ul').on('click', 'li', {name:'hello'}, function(){
    $(this).css('background', 'red');
})
```

```
$('#input').focus(function(){
    $(this).css('background', 'red');
});
//$('#input').trigger('focus');
$('#input').triggerHandler('focus'); //会触发响应的事件，但是不会触发默认的行为
```

```
<div id = "div1"></div>
<span id="span1"></span>
$('#div1').on('click', function(){
	alert(1);
	$('#div1').off('click');
});
$('#div1').on('click', function(){
	alert(2);
});
$('#div1').trigger('click'); //主动触发事件
```

通过原生的实现上面的，简单的模拟实现：
```
var oDiv = document.getElementById('div1');
var oSpan = document.getElementById('span1');
var aaa = function(){
	alert(1);
};
var bbb = function(){
	alert(2);
};
//add(oDiv, 'click', aaa);
//remove(oDiv, 'click', aaa);
//自定义事件
add(oSpan, 'show', aaa);
add(oSpan, 'show', bbb);
trigger(oSpan, 'show');

function add(obj , types, fn){
	obj.listeners = obj.listeners || {}; //映射不同的元素 4353
	obj.listeners[types] = obj.listeners[types] || [];  //定义不同元素的数组4402
	obj.listeners[types].push(fn); //4426
	obj.addEventListener(types, fn, false); //4409
}
function remove(types, fn, false){
	obj.removeEventListener(types, fn, false);  //4494

	delete obj.listeners[types];  //4497
}
function trigger(obj, types){
	var arr =  obj.listeners[types] || [];
	for(var i=0; i<arr.length; i++){  //4585
		arr[i]();
	}
}
```

事件中所有的数据都是挂在data_prev对象上。

关于这个挂载的对象，可以在jquery中console一下：
```
var elemData = {
	events : {
		'click' : [  //这个是一个数组，arr，数组下有两个属性 arr.delegateCount（委托计数），arr.length
		//click 绑了2次，就会有2个对象
			{

			},
			{}
		],
		'mouseover' : [
			{}
		]
	},
	handle : function(e) {

	}
}
```
当使用了委托之后，以上的数据会缓存到其他身上。而非被元素身上。比如例子中是会缓存到body身上。

（5029）on data用于传参的处理 一般type都是字符串，像是click或者onmouseover， if里面处理这样的情况： `$('#div1').on({` `'click' : function(){` `alert(123);` `},` `'mouseover' : function(){` `alert(456);` `}` `})` 处理参数顺序的兼容问题。	`jQuery.fn.extend({` `on: function( types, selector, data, fn, /*INTERNAL*/ one ) {` `var origFn, type;` `// Types can be a map of types/handlers` `if ( typeof types === "object" ) {` `// ( types-Object, selector, data )` `if ( typeof selector !== "string" ) {` `// ( types-Object, data )` `data = data || selector;` `selector = undefined;` `}` `for ( type in types ) {` `this.on( type, selector, data, types[ type ], one );` `}` `return this;` `}` `if ( data == null && fn == null ) {` `// ( types, fn )` `fn = selector;` `data = selector = undefined;` `} else if ( fn == null ) {` `if ( typeof selector === "string" ) {` `// ( types, selector, fn )` `fn = data;`

	data = undefined; } else { // (types, data, fn) fn = data; data = selector; selector = undefined; } } if (fn === false) { fn = returnFalse; } else if (!fn) { return this; }
针对one那个方法的，让事件只执行一次。 $('#div').one('click', function(){ alert(123); }); 一进来就取消掉。 通过apply来调用函数。 不管fn调用一次还是多次，都要绑定guid标识。 这里并没有实质性的操作，大部分是参数的维护处理	if (one === 1) { origFn = fn; fn = function(event) { // Can use an empty set, since event contains the info jQuery().off(event); return origFn.apply(this, arguments); }; // Use same guid so caller can remove using origFn fn.guid = origFn.guid || (origFn.guid = jQuery.guid++); } return this.each(function() { jQuery.event.add(this, types, fn, data, selector); }); },
（5082）one 让事件只执行一次的操作	one: function(types, selector, data, fn) { return this.on(types, selector, data, fn, 1); },
（5085）off 这里是参数的修正 给每一个元素都调用remove操作	off: function(types, selector, fn) { var handleObj, type; if (types && types.preventDefault && types.handleObj) { // (event) dispatched jQuery.Event handleObj = types.handleObj; jQuery(types.delegateTarget).off(handleObj.namespace ? handleObj.origType + "." + handleObj.namespace : handleObj.origType, handleObj.selector, handleObj.handler); return this; } if (typeof types === "object") { // (types-object [, selector]) for (type in types) { this.off(type, selector, types[type]); } return this; } if (selector === false || typeof selector === "function") { // (types [, fn]) fn = selector; selector = undefined; } if (fn === false) { fn = returnFalse; } return this.each(function() { jQuery.event.remove(this, types, fn, selector); }); },

==========

==========

==========

==========

==========

==========

==========

//第11部分
miaov 51
attr prop val addClass

jQuery.fn.extend({
    attr
    removeAttr
    prop
    removeProp
    addClass
    removeClass
    toggleClass 有删除 没有添加
    hasClass
    val
})

jQuery.extend({
    valHooks
    attr
    removeAttr
    attrHooks
    propFix
    prop
    propHooks
})
attr 可以识别自定义属性
prop 不能识别自定义属性
attrhooks 具体检测兼容的处理

//第12部分 事件操作
miaov 59
jQuery.event = {
    global 事件的全局属性，jquery还没用到
    add
    remove
    trigger
    dispatch 派发事件的具体操作
    handlers 函数执行顺序的操作
    props JQ中共享原生JS的event属性
    fixHooks 具体的event兼容
    keyHooks 键盘的event兼容
    mouseHooks 鼠标的event兼容
    fix event对象的兼容处理
    special 特殊事件的处理
    simulate focusin
}
jQuery.Event = function(){
    isDefaultPrevented
    isPropagationStopped
    isImmediatePropagationStopped
    preventDefault
    stopPropagation
    stopImmediaPropagation
}

jQuery.fn.extend(
    on
    one
    off
    trigger
    ttiggerHandler 不会触发事件的默认行为
)

//6720
.click();
mouseover();
jQuery.fn.extend({
    hover
    bind
    unbind
    delegate
    undelegate
})

miaov 64
needsContext 和委托有关，伪类的情况下为真
origType 原始类型，当初想要绑 的事件，可能不兼容
type 现在类型，已被修改成的兼容类型

miaov 65
special 特殊事件的处理 onmouseenter,focusin
add中调用的是dispatch
委托的事件会被先执行

miaov 67
键盘值兼容

miaov 68
特殊事件的处理
jQuery.event.special
load
focus 不支持冒泡
blur
click
beforeunload
mouseenter
mouseleave
focusin
focusout

miaov 71

//DOM操作
miaov 72-76
$.fn.extend({
    find
    has 在子项上看是否有满足条件的
    not
    filter
    is
    closest
    index
    add
    addBack
})
miaov77
jQuery.each({
    parent
    parents
    parentsUntil
    next
    prev
    nextAll
    prevAll
    nextUntil
    prevUntil
    sibling
    children
    contents
})
jQuery.extend({
    filter
    dir
    sibling
})
miaov 79
jQuery.fn.extend({
    text
    append
    before
    after
    remove 事件全部删除，添不回去
    empty
    clone
    html
    replaceWith
    detach 事件会保留，可以添回去
    domManip
})
miaov 80
text empty clone
miaov 82
html
append
preappend
before
after

miaov 84
文档碎片
replaceWith

miaov 85
append appendTo
包装的操作wrap
unwarp 除了body之外的父级会被删掉

// 下一个部分 css 样式操作
86~90

miaov 90
show
hide
toggle

miaov 92

miaov 94