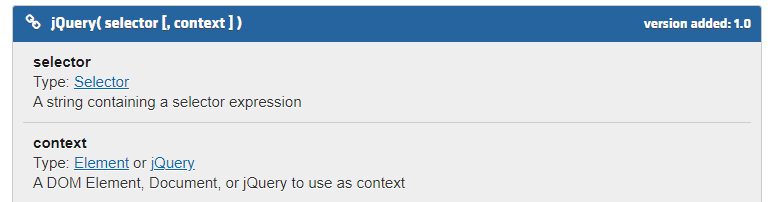
那些年我们一起追过的jQuery ---- 2. 选择器（一）

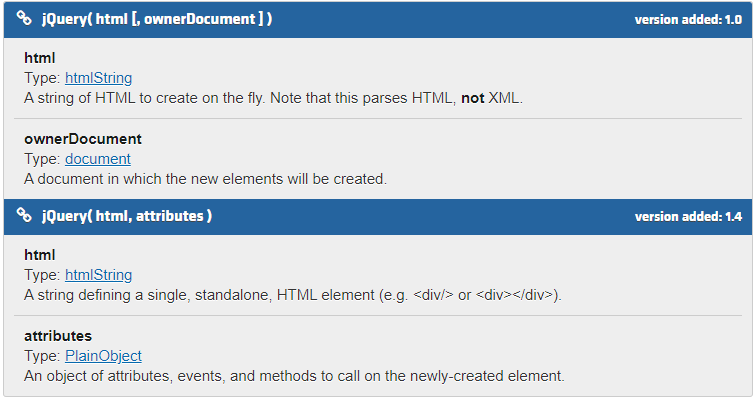
紧接上篇文章整体架构剩下的jQueryAPI中if ( typeof selector === "string" )的case，由此引出jQuery选择器中的相关解析。本章节介绍了jQuery()API中9中用法中的：jQuery( selector [, context ] )查询文档元素API的实现。和jQuery( html [, ownerDocument ] )，jQuery( html, attributes ) 动态创建由 jQuery 对象包装的 DOM 元素及相关工具方法的实现。并简单介绍了Sizzle（一个纯粹的javascript css选择器引擎）。

1. if ( typeof selector === "string" )

上篇文章说到jQuery.fn.init = function( selector, context, root ) {} 用于生成jQuery对象。根据参数重载有多种不同使用方式。上篇已经分析了4中，还剩一种当selector是字符串的情况if ( typeof selector === "string" )将用于实现三个API。

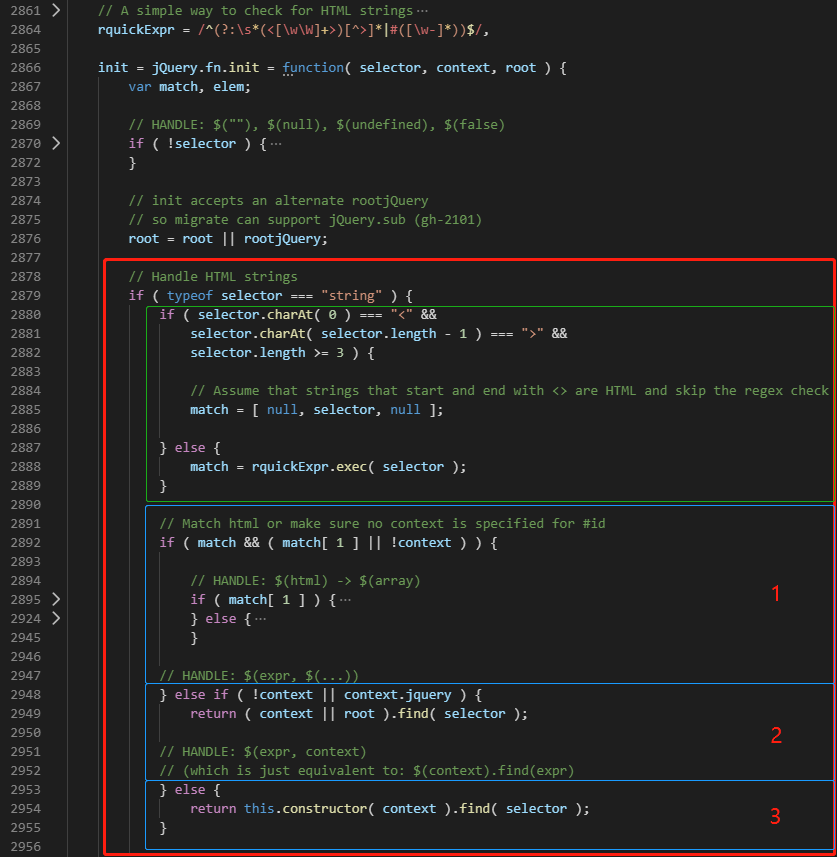


接受一个包含一个CSS选择器的字符串，用于匹配元素，



根据提供的原始 HTML 标记字符串，动态创建由 jQuery 对象包装的 DOM 元素，并可附带属性及事件。

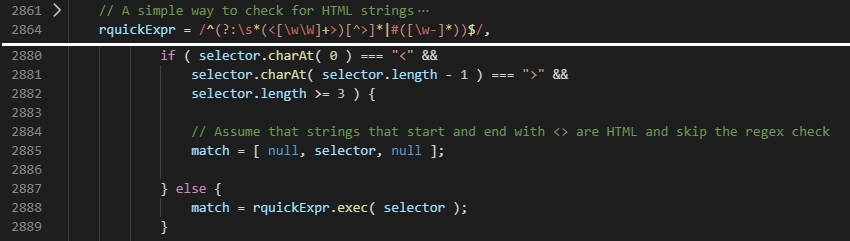
从源码开始一窥究竟：



源码试析：

因为$(‘selectorStr’)有两种使用方式。一种是查询元素例如：$(‘.container’)，一种是创建元素$(‘<div>’)，所以需要一个标识，也就是图示中绿框中的match，如果匹配到selectorStr中有完整的标签元素就用来生成元素，没有匹配到就用来查找元素。

细节：



如果selectorStr匹配到了< 开始，> 结束并且字符串内不只<>，长度大于等于3就视为一个完整标签。当然为了健壮性还需要正则表达式进行匹配是否存在标签，不知道能不能看懂下面正则表达式的行为：

rquickExpr = /^(?:\s\*(<[\w\W]+>)[^>]\*|#([\w-]\*))$/,

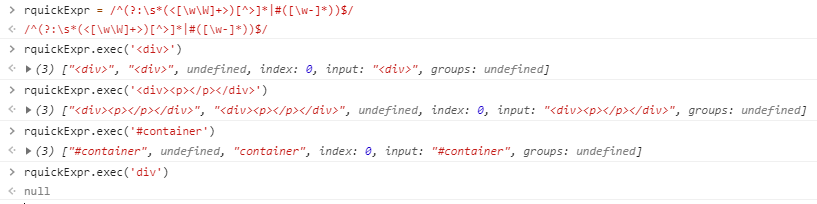
推荐一篇正则入门文章：<https://www.cnblogs.com/yang-C-J/p/6263066.html>。

rquickExpr被一个（）包裹并且有?:修饰意味着此（）不作为分组捕获信息。中间 还有优先级最低的或 | 分割，所以这个正则可以看做是两部分：

1. \s\*(<[\w\W]+>)[^>]\* // 匹配标签 并且 捕获标签名tagName
2. #([\w-]\*) // 匹配id 并且捕获id名

rquickExpr.exec('selectorStr')

如果 exec() 找到了匹配的文本，则返回一个结果数组。否则，返回 null。此数组的第 0 个元素是与正则表达式相匹配的文本，第 1 个元素是与 rquickExpr 的第 1 个子表达式（括号分组）相匹配的文本（如果有的话），第 2 个元素是与 rquickExpr 的第 2 个子表达式相匹配的文本（如果有的话），以此类推。除了数组元素和 length 属性之外，exec() 方法还返回两个属性。index 属性声明的是匹配文本的第一个字符的位置。input 属性则存放的是被检索的字符串 string



由于存在（）分组即生成了子表达式，exec匹配到的结果match[0]是selectorStr字符串，match[1]为所匹配到的标签名，match[2]为所匹配到的id名。



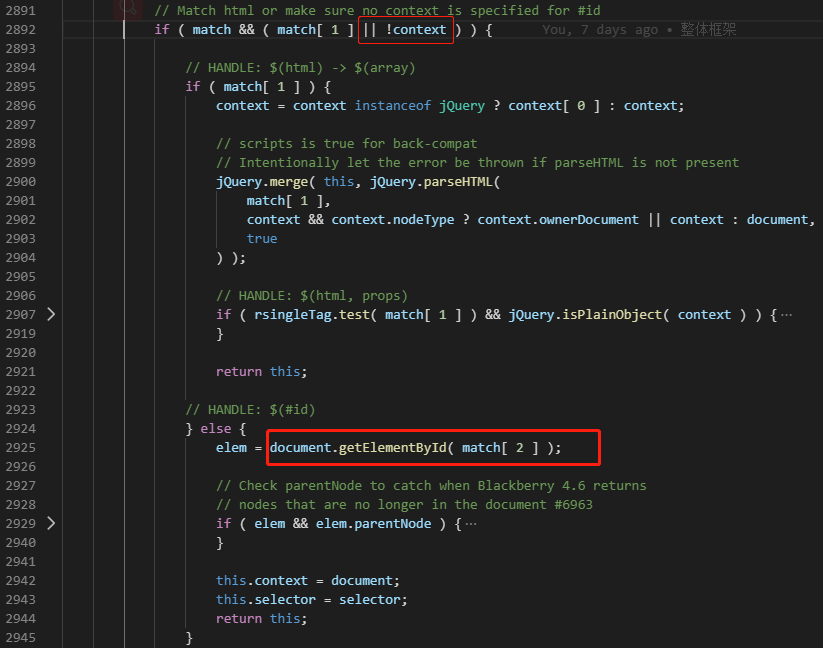
分支结构很明显了，

1. 当有匹配到标签或者id时，match有值。

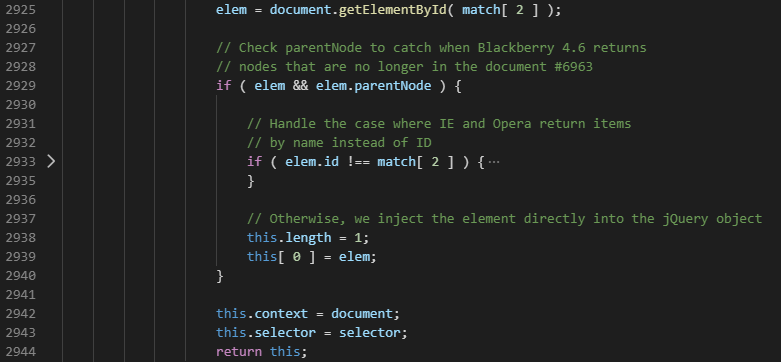
走if ( match && ( match[ 1 ] || !context ) )分支，里面又有两个分支分别：

* 1. match[1]有值时，意味着匹配到标签（例：<div>）的情况
  2. match[2]有值时，意味着匹配到id的情况

注意：



因为这里匹配到id时是从document上寻找，所有判断条件中加入了||!context调用$(selector,[context])时不能有context的限制。



匹配到id后就会把它包装成jQuery对象返回。

回过头来看match[1]有值时，匹配标签的情况



context = context instanceof jQuery ? context[ 0 ] : context;

确定上下文元素，以便得到文档对象，createElement()需要由文档对象调用。

jQuery.merge( this, jQuery.parseHTML(

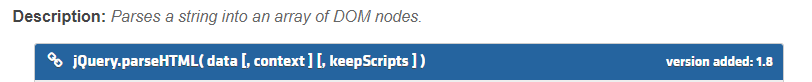
match[ 1 ],

context && context.nodeType ? context.ownerDocument || context : document,

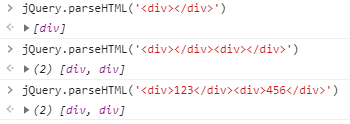
true

) );

合并dom数组到this对象



将字符串解析成一个个DOM节点并放置到数组中



jQuery.parseHTML(

match[ 1 ],// 匹配的标签字符串

context && context.nodeType ? context.ownerDocument || context : document,

// 无context 传入document，有context 传入context 所属的文档对象ownerDocument

true

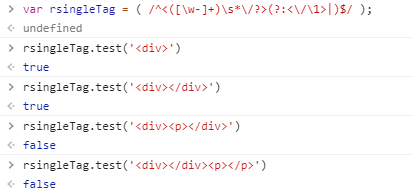
) );

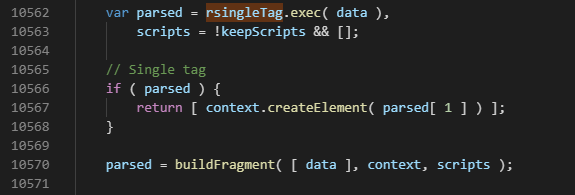
看一下jQuery.parseHTML内部如何封装



rsingleTag 正则用于判断标签字符串是否为单标签 例： ‘<div>’ ‘<div></div>’ 而非多标签

‘<div></div><div></div>’或嵌套’<div><span></span></div>’

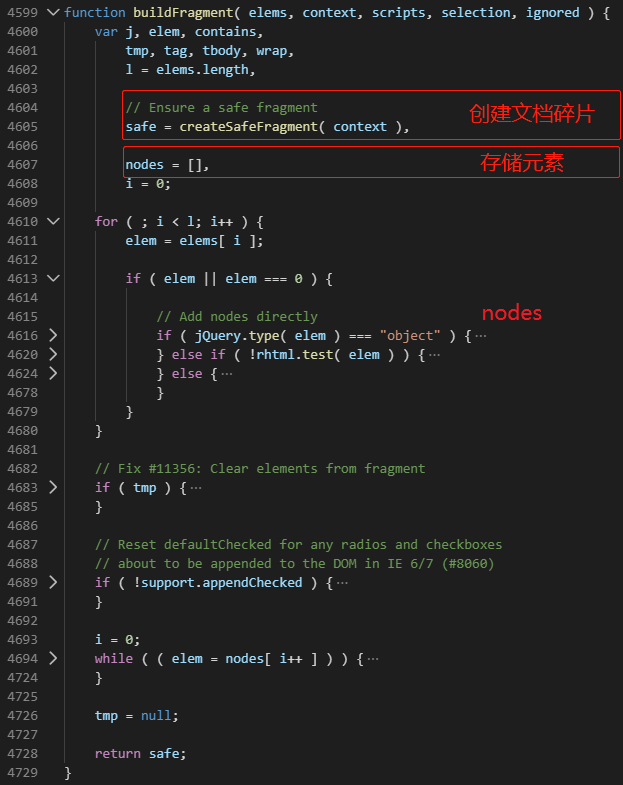




如果只有一个标签就直接使用 例：document.createElement(‘div’) 创建元素并包裹成数组返回。

如果不是单标签，是多标签的话。需要使用buildFragment() 创建文档碎片，简单解释一下：就是创建一个文档碎片存储在内存中临时存储需要创建的元素。最后一次性加入到页面文档中，可以提高性能。

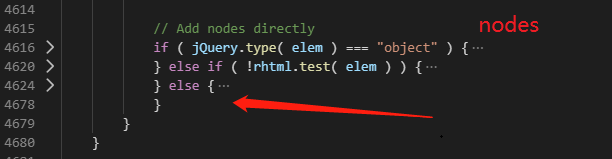
来看下buildFragment()方法里封装了什么内容



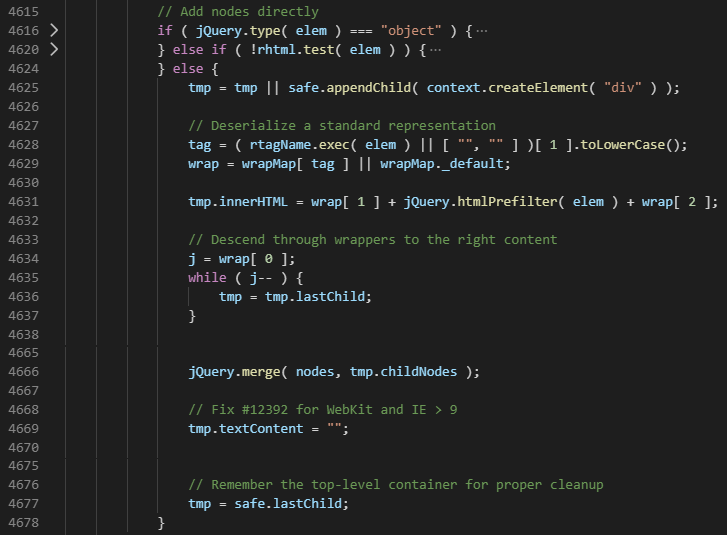
createSafeFragment()方法创建一个安全的文档碎片结构：添加了一些不支持html5语义化标签的处理。



因为处理的是标签字符串，所以nodes处理阶段走最后一个分支



展开来看：



1. 首先在文档碎片中加入一个新创建的div元素用于包裹匹配的标签创建。

tmp = tmp || safe.appendChild( context.createElement( "div" ) );

tmp 为 刚创建的 div元素

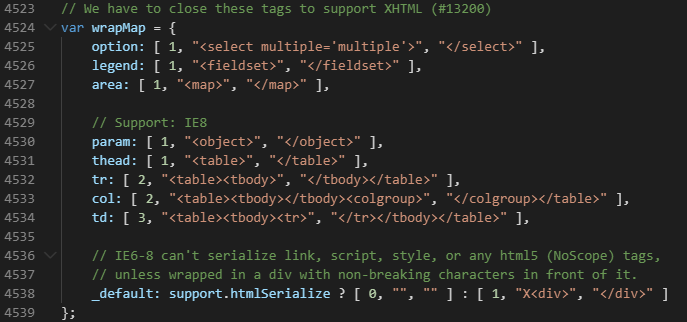
1. 获取最外层标签名tagName 存储到 tag中

tag = ( rtagName.exec( elem ) || [ "", "" ] )[ 1 ].toLowerCase();

1. wrap包裹元素



先来看看wrapMap里是一个什么对象



Key：为标签名。value：是一个数组，数组中第二个值和第三个值正是嵌套key标签的父标签，数组中第一个值是嵌套的深度。

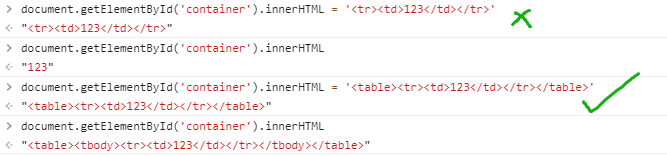
为什么要这么做：

因为：

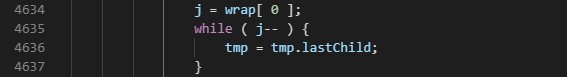
tmp.innerHTML = wrap[ 1 ] + jQuery.htmlPrefilter( elem ) + wrap[ 2 ];

用innerHTML添加元素字符串时，会先解析元素字符串是否符合元素包裹规则

如果不符合是inner不了的



以<tr><td>123</td></tr> 字符串为例，包裹<table><tbody></tbody></table>后tmp下第一个元素就变更为table，所以还需要找回最初的元素标签tr



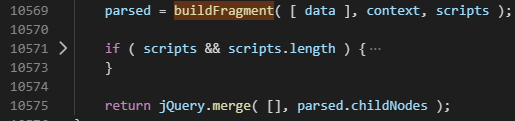
1. 把创建的元素添加到nodes存储



1. 最后清空safe文档碎片，然后遍历nodes元素只添加符合selectorStr创建的元素到safe文档碎片。返回safe

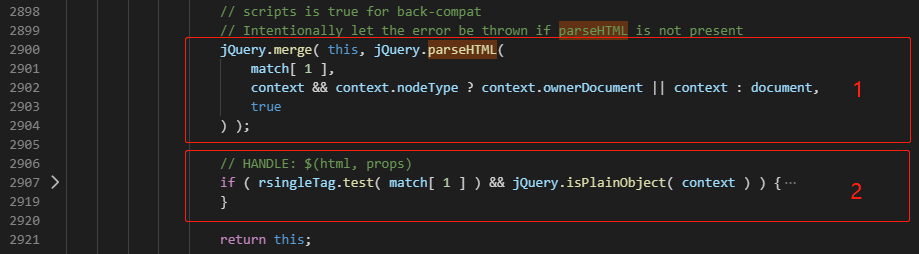


回到jQuery.parseHTML 转换HTML方法内：

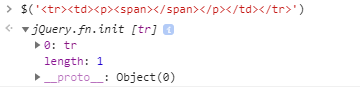


buildFragment()得到了文档碎片包含了元素集合，被直接返回；

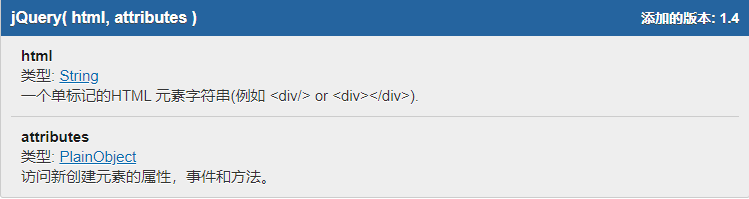
再回溯到jQuery.fn.init()方法中

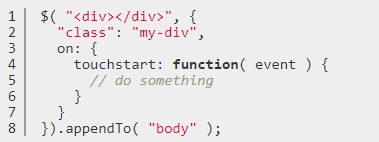


没有2的情况下，直接把元素合并到jQuery对象上

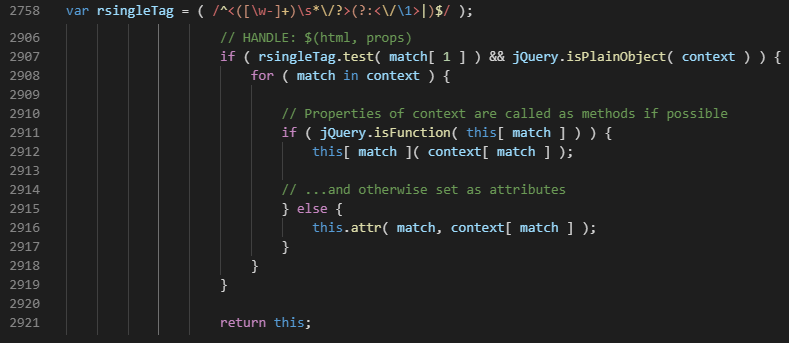


为2的条件是在判断selectorStr为单标签，并且第二个参数是个简单对象，可能是用来为标签添加属性或者事件。



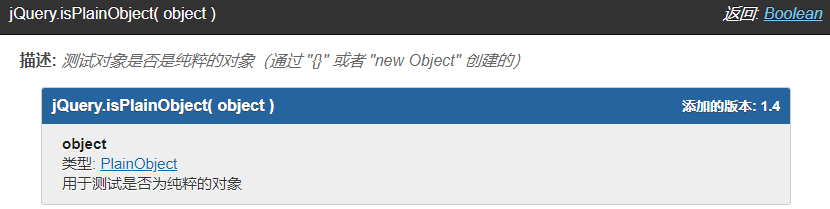


代码展开

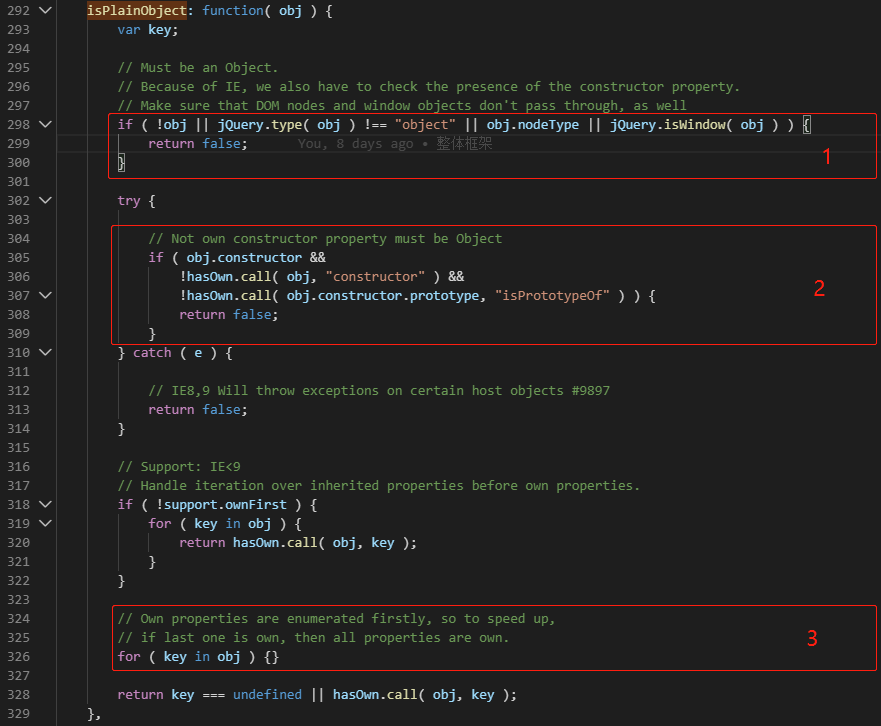


rsingleTag.test( match[ 1 ] ) // 检测match[1]是否为单标签

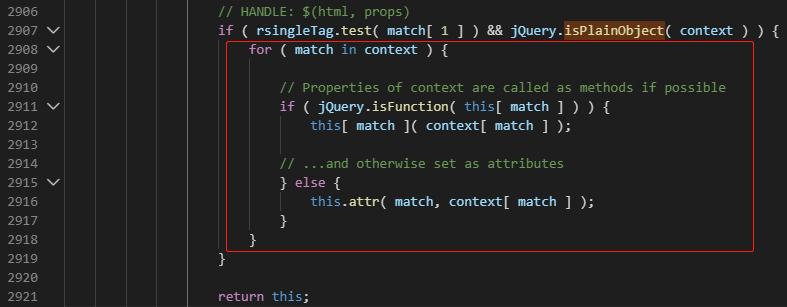
jQuery.isPlainObject( context ) // 检测context是否为简单对象



查看一下isPlainObject的代码



1. 先剔除（非对象object）和ElementNode，window对象的情况
2. 然后判断是否为简单对象：由{}字面量形式或new Object 创建的对象
3. 判断所有属性都是自己的，如果不是说明是有继承来的属性，则不是简单对象



遍历简单对象上的属性添加到jQuery对象上，然后返回jQuery对象

例如简单对象：

    {

        "class": "my-div",

        "on": {

            touchstart: function (event) {

                // Do something

            }

        }

    }

$("<div></div>", {

    "class": "my-div",

    "on": {

        touchstart: function (event) {

            // Do something

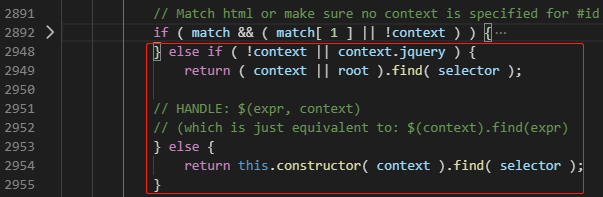
        }

    }

}).appendTo("body");

创建标签元素和匹配简单#id的模式结束

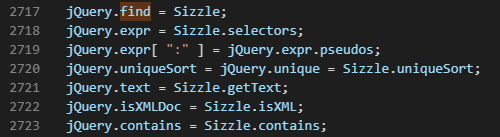
1. 复杂jQuery选择器的情况



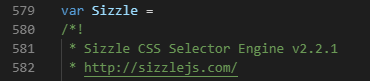
两个分支其实是一个形式无非是如果是jQuery对象直接调用find，非jQuery对象包装成jQuery对象在调用find。

$( "div.foo" );

来看下find方法



来自Sizzle，这说明玩意啊！



一个css选择器引擎，嗯。。。。。。。



下节再讲吧！ to be continued