那些年我们追过的jQuery ----- 第一节 整体架构

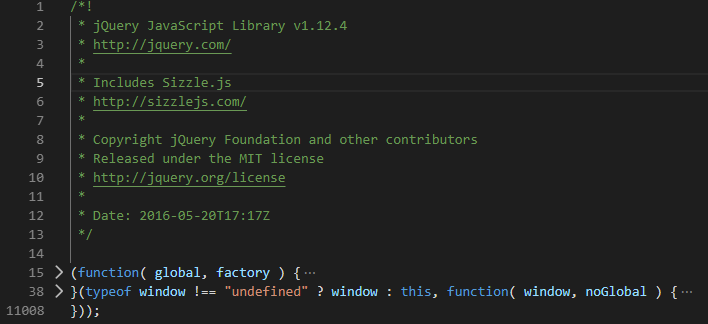
1. 选取jQuery版本



现在jQuery已经迭代了70个版本，分出了1,2,3个大版本

* 1.x：兼容ie678,使用最为广泛的，官方只做BUG维护，功能不再新增。因此一般项目来说，使用1.x版本就可以了，最终版本：1.12.4 (2016年5月20日)
* 2.x：不兼容ie678，很少有人使用，官方只做BUG维护，功能不再新增。如果不考虑兼容低版本的浏览器可以使用2.x，最终版本：2.2.4 (2016年5月20日)
* 3.x：不兼容ie678，只支持最新的浏览器。除非特殊要求，建议不使用3.x版本的，很多老的jQuery插件不支持这个版本。目前该版本是官方主要更新维护的版本。

这里选择1.12.4版本的jQuery进行分析。



11008行代码，嗯好长啊，初看代码，感觉无从下手的感觉。

1. 怎样阅读源码？

思考一下问题：

* 为什么要阅读源码？（答：面试官会问… 请出去，好吗）

本人是看到一个框架或者说工具给出的解决方案好用，或者API有独到惊艳的地方，迫使我情不自禁的想知道它是怎么做到的，怎么实现的。不知道你们是不是。（标准答案哦！）

既然是被用法和API所吸引，就应该逐个击破，化整为零，毕竟源码太长，着重点先后理解。

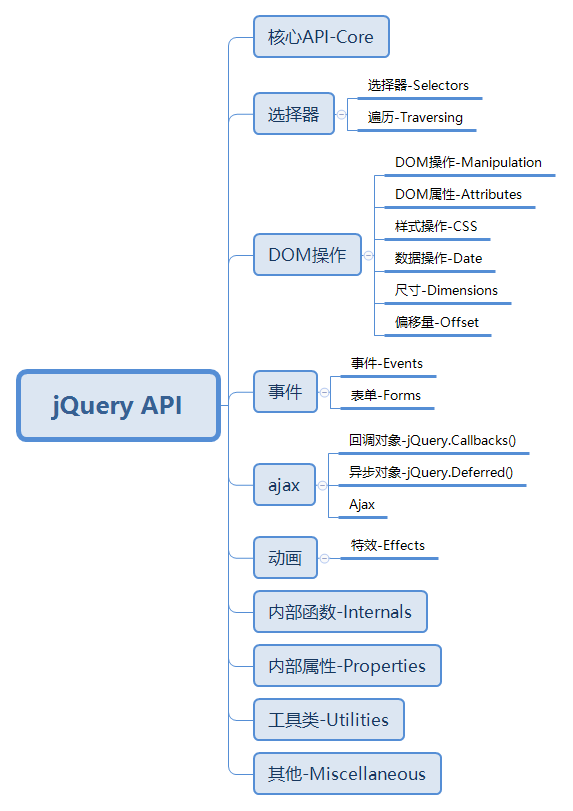
三、从API入手jQuery源码

<https://api.jquery.com/> （英文API网站）

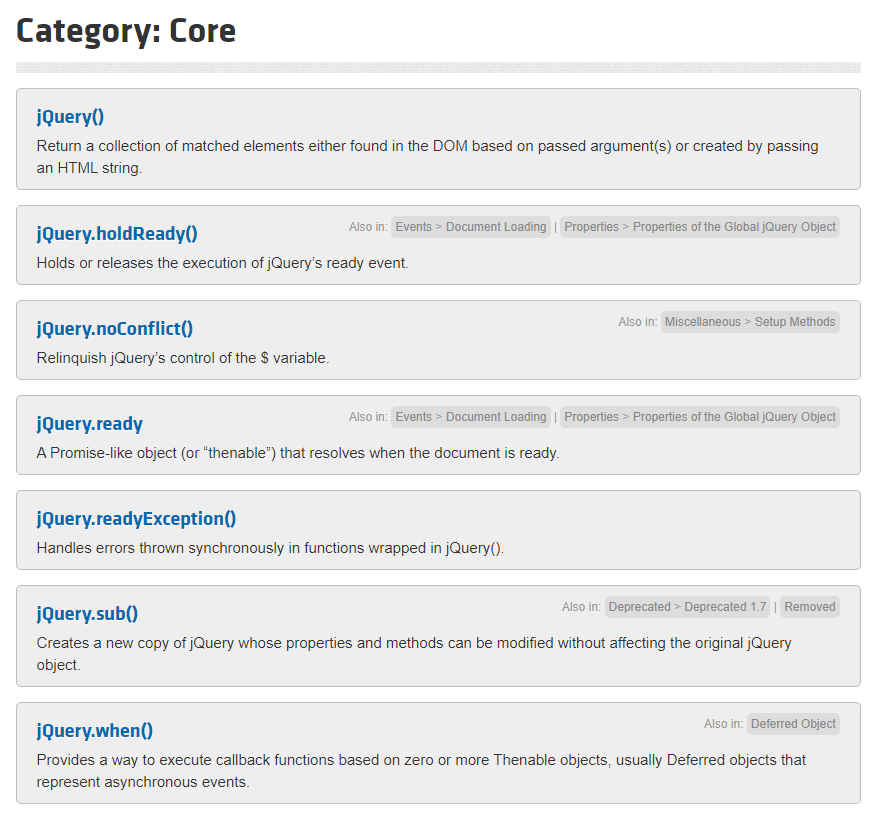
<https://www.jquery123.com/> （中文API网站，如果有翻译理解不了的，还可以对照英文自行翻译理解）

官网中API 分了11个类别，

我划分了一下大类：



今天先从核心API看起，



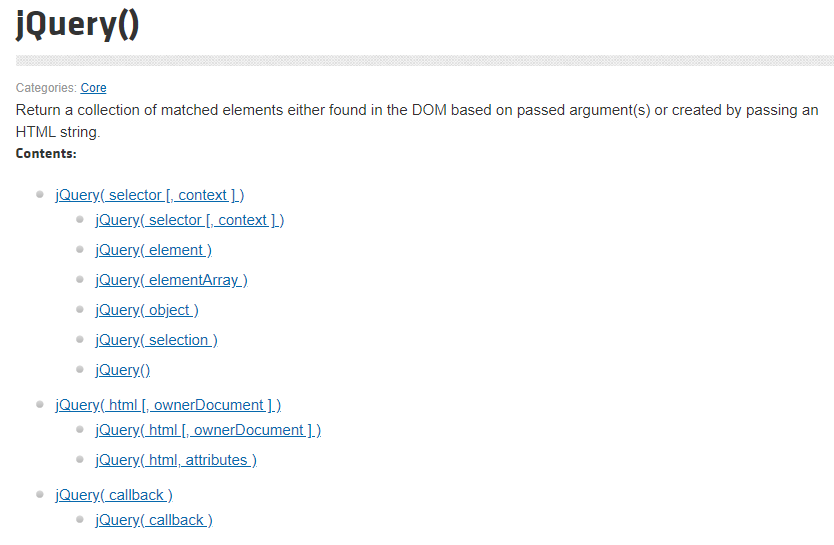
jQuery.sub()已经在1.7中被移除，不在欣赏了，

jQuery.holdReady()和jQuery.ready()在事件的文档加载中有更详细的应用场景，在哪里在做欣赏，

jQuery.when() 在异步对象中应用，留在那里这欣赏。

嗯，还剩下，jQuery()和jQuery.noConflict() ，其中jQuery()也是最核心的API，没有这个方法，就没有jquery对象，怎么操作DOM，样式，和动画，及事件呢？

核心解析jQuery()

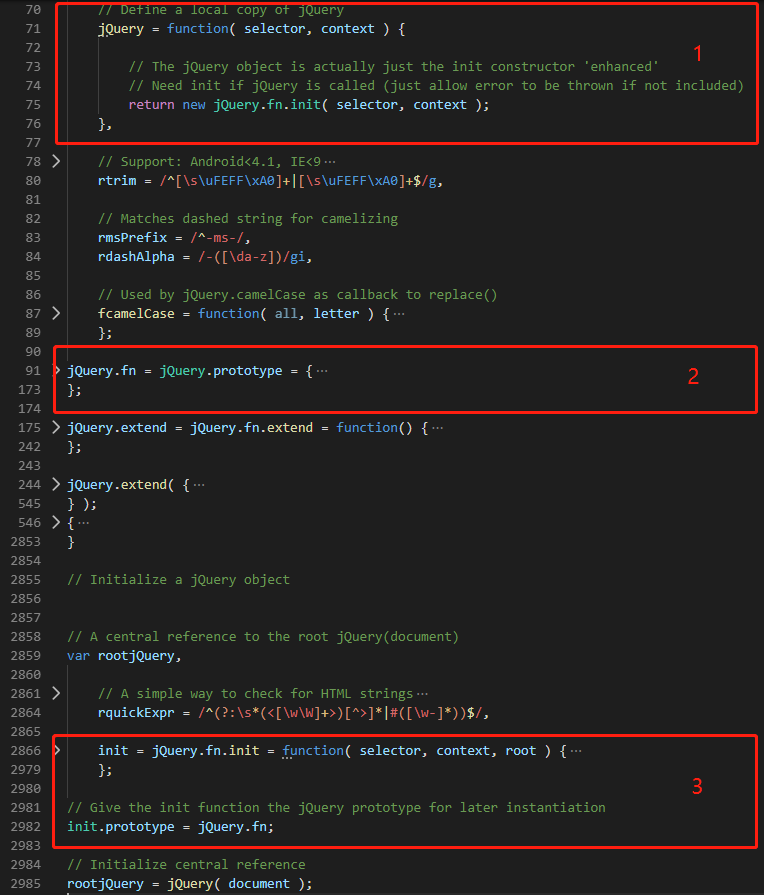


jQuery()方法有9种重载方式，好多啊，一反我们平时开发倡导的单一职责，但对一个工具类或者框架来说确实用户最容易接受的。用户不需要记太多的方法，一个方法处理一类甚至多类相近功能。这需要jQuery有强大的封装变换的能力，经历了这么多版本迭代。jquery()方法内部逻辑已经相当清楚

1.来看下内部的具体实现



jquery() 会返回一个jquery.fn.init对象



源码实现：

第一步：

调用jquery()方法返回一个jQuery.fn.init对象。因为return 了一个init对象，即使new jquery()也是返回init对象。（构造器方法如果有显示的return 一个对象，new 构造器 就方法这个对象而不在是this实例）

第二步：

jQuery.fn 的引用 指向jQuery.prototype 。感觉就是起了一个别名

第三部：

jQuery.fn.init.prototype = jQuery.fn = jQuery.prototype

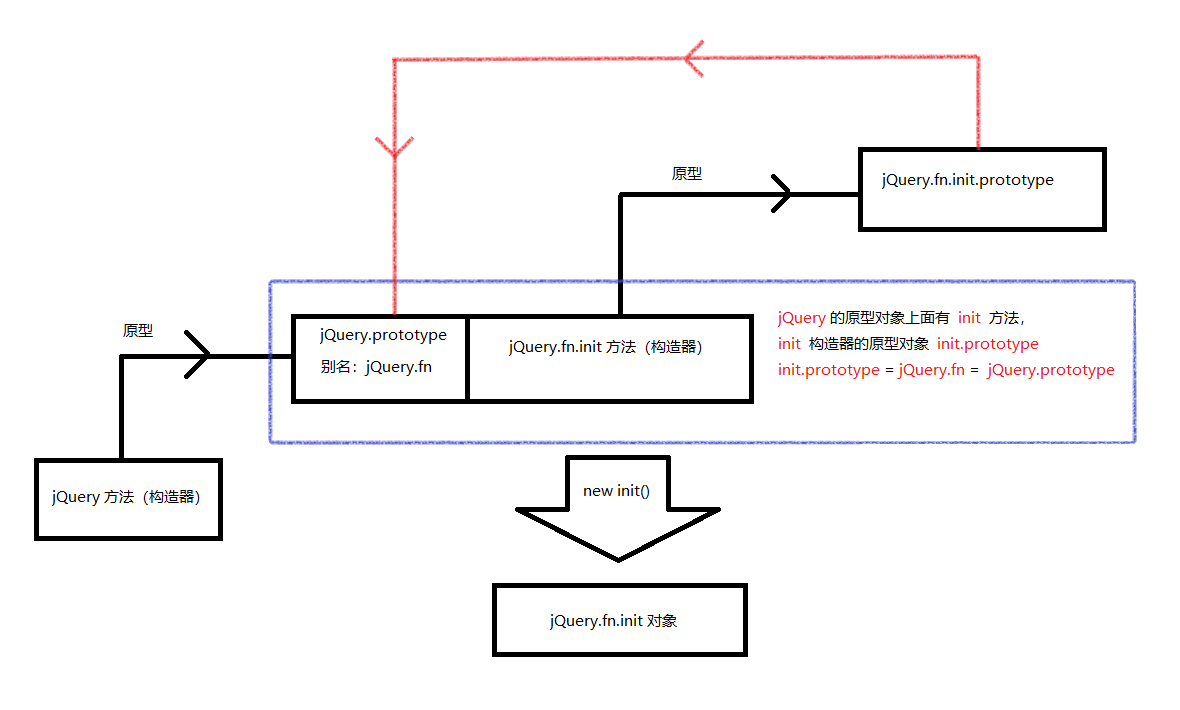
最后结果就是，创建出来的jQuery.fn.init 对象的原型（\_\_proto\_\_） 指向了 jQuery.prototype。

这样做的效果：

使用了jQuery工厂创建了一个jQuery.fn.init对象，

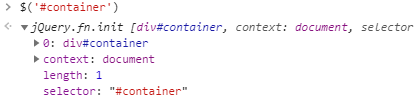
因为jQuery.fn.init对象的原型指向jQuery工厂的原型，

所以jQuery.fn.init对象还可以调用 jQuery.prototype 上的方法



应该就是这个样子，原型链很熟悉的同学麻烦看看有么有错误。

jQuery.fn.init构造器一览



Init 对象会是一个类数组，因为匹配到的元素可能是一个也可能是多个，所以需要象数组一样存储这些dom，还添加两个自身属性context 上下文（在哪个元素内进行查旭匹配），selector（查询参数）

我们来看一下init构造器到底如何处理的封装：



init 中有5个关键的判断分支

1. if ( !selector )

处理 $(""), $(null), $(undefined), $(false) 的情况

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

处理正常 选择器行为 $("p") $("#test") $(".test") $("p.intro")

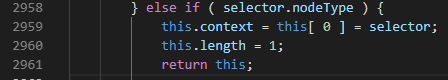
这种情况比较复杂，留再选择器中再试析

1. if ( selector.nodeType )

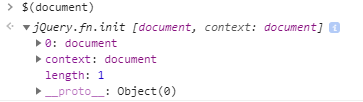
处理 包装普通dom元素的行为

$(document)

$(document.getElementById("name"))



为 jQuery.fn.init对象添加 length，context，[0]，包装成jQuery对象返回



1. if ( jQuery.isFunction( selector ) )

包装文档就绪事件

$(function(){

// 开始写 jQuery 代码...

});

if ( jQuery.isFunction( selector ) ) {

            return typeof root.ready !== "undefined" ?

// root = $(document)

                root.ready( selector ) :

// $(document).ready(function(){})

                // Execute immediately if ready is not present

                selector( jQuery );

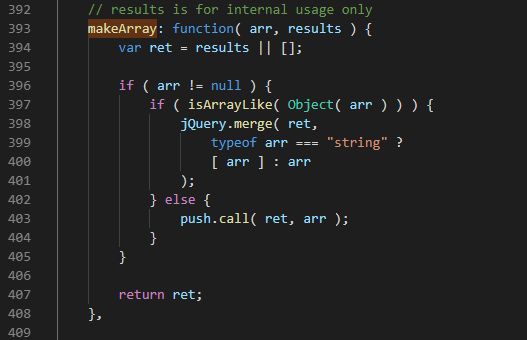
        }

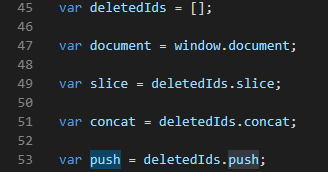
1. return jQuery.makeArray(selector,this)

如果前四种没有匹配上，就返回一个init()构造器包裹的类数组，为什么是类数组，因为对象有length属性和下标并且length可读写，但原型不指向Array.prototype, 它指向jQuery.prototype



查看jQuery.makeArray(selector,this) 做了那些事情



push == Array.prototype.push

判断arr即传入的selector是否是类数组，

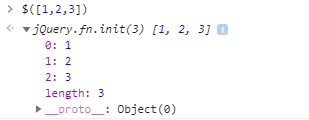
如果是就遍历元素添加到ret中，

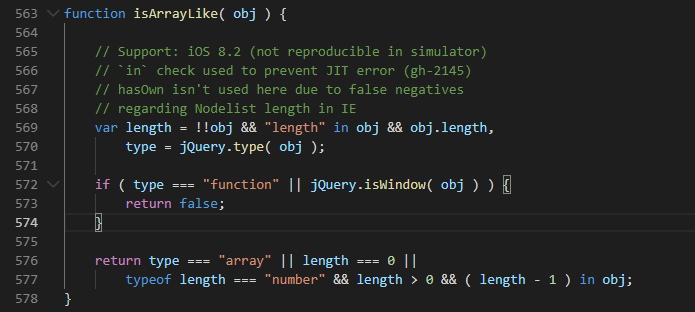
如果不是就整个arr添加多ret中

返回 ret 即为传入的this （ new jQuery.fn.init() ）上加入的length和 arr值

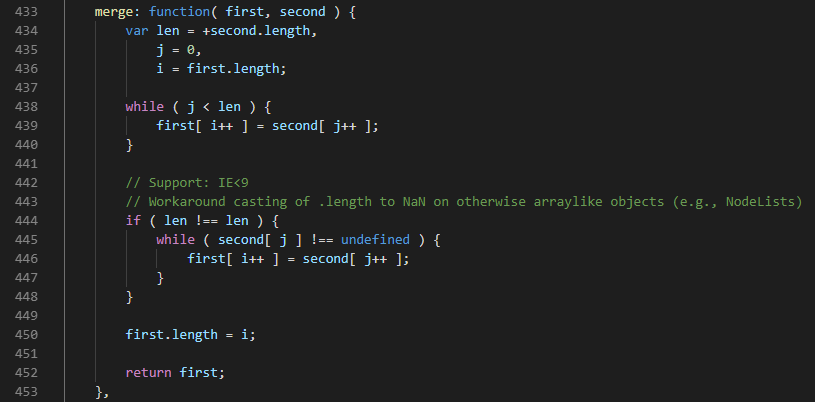
例：







注意：判断是否为类数组，需要把function排除，虽然一个函数对象有length属性但length为参数数量且不可写，所以不能使用Array.prototype上的方法就不算类数组。



second对象上的元素添加到first对象上



这种情况留在选择器在试析

# 二、jQuery.noConflict()

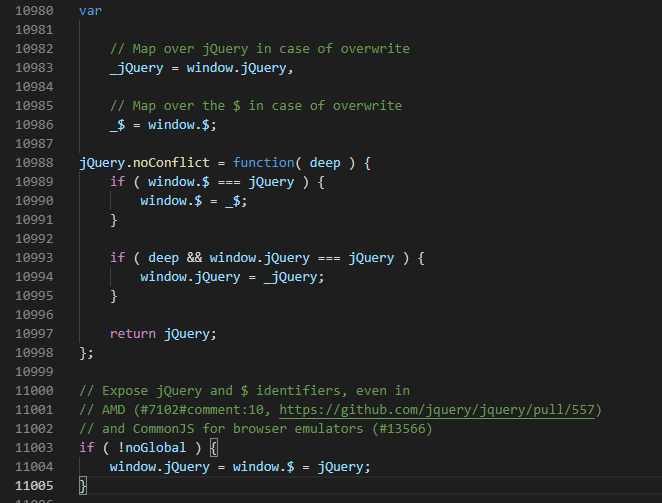
**jQuery.noConflict( [removeAll ] )**

* **removeAll**

类型: [Boolean](https://www.jquery123.com/Types/#Boolean)

一个布尔值，判断是否从全局作用域中内去除所有jQuery变量(包括jQuery本身)。

很多javascript库使用 $ 作为一个函数或者变量名，正如jquery做的一样。在jQuery下  $只是jQuery的一个别名，所以不使用 $所有功能都是有效的。如果我们需要同时使用jQuery和其他javascript库，我们可以使用 $.noConflict()把 $的控制权交给其他库。旧引用的$ 被保存在jQuery的初始化; noConflict() 简单的恢复它们。



    // Map over jQuery in case of overwrite

    \_jQuery = window.jQuery,

    // Map over the $ in case of overwrite

    \_$ = window.$;

jQuery 已经保存了原本window上的jquery对象到\_jquery, $对象到\_$,

随意我们只需要调用jQuery.noConflict

方法就能 还原原本$或jQuery 对象的指向。

比如我先加载了 jquery1.8 版本，现在window.jquery和window.$都保存的是1.8版本对象。

再 加载jquery1.9版本，就会覆盖原本window上的jquery和$对象。但并没有丢失1.8版本的jquery和$ 他们被保存在了\_jquery和\_$内 可以随时找回。