JQ（简介)选择器

write less, do more

jQuery历史：

jQuery是一个JavaScript类库，它通过封装原生的JavaScript函数得到一整套定义好的方 法。它的作者是John Resig，于2006年创建的一个开源项目，随着越来越多开发者的加入， jQuery已经集成了JavaScript、CSS、DOM和Ajax于一体的强大功能。它可以用最少的代码， 完成更多复杂而困难的功能，从而得到了开发者的青睐。

jQuery是一个类库：

类库就是一个综合性的面向对象的可重用的类型集合。 这些类型包括接口、抽象类和具体类。

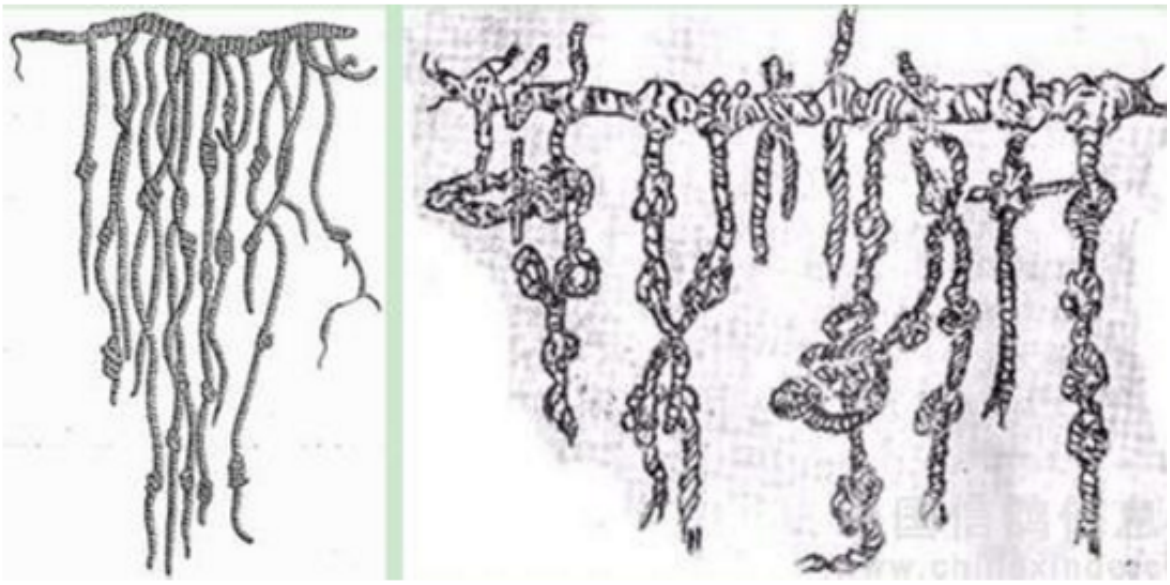
接口的最主要的作用是达到统一访问，就是在创建对象的时候用接口创建， 【接口名】 【对象名】=new 【实现接口的类】，这样你像用哪个类的对象就可以new哪个对象了，不需要改原来的代码，就和你的USB接口一样，插什么读什么，就是这个原理。

抽象类定义借口。

一句话总结类库。 就是JavaScript的工具库。 原始社会生火方式，

钻木取火。 但是有了火柴就不一样了。

以前没有数字的时候用结绳记事， 有了文字之后就不一样了。



“哇，艾伦，你看那边的女孩子，好可爱哦。。”听了阿明的话后，艾伦看了看那边，“果然……很美丽呐”

三笠默默的看着，心中划过一丝不屑。

兵长拉着女生的手走了进来，清了清嗓子说：“这是……”

“夏微凉。”女生抢先回答了。这个女生真的好漂亮……樱色及腰的长发，以及明亮的水蓝色的瞳孔。

兵长意识到任务在身，但是……为什么……自己的手……离不开她的手呢……

可恶……这种奇怪的念头……

“等我回来。”兵长在微凉耳边悄悄说，然后哈了一口气。夏微凉脸“倏”地红了，自

以前的JavaScript需要些很多的兼容，写很多重用性高的方法。但是有了jquery之后就不一样了。

jquery都包括什么 JavaScript， CSS，DOM ，AJAX。

开发前准备

html，css以及JavaScript知识

1.jquery-1.XX.X.js

 普通文件

实例1.建立jquery环境。引用一个外部的jquery库。和引入外部script方法一致。

jquery的第二种引入方式，CND引入。

<script src="http://libs.baidu.com/jquery/2.0.0/jquery.js">
</script>

jQuery1.XX版本和2.XX版本的区别

1.XX版本兼容IE 6， 7 ， 8 2.XX版本放弃了对IE6，7 ，8 的兼容。

百度CDN。

在产品上线时推荐使用线上CDN，因为线上CDN可以从浏览器缓存中加载。加快页面加载速度。目前最好的并且通用的CDN就是百度CDN。 谷歌的CDN会被墙。

CDN是什么?

内容分发网络，是一种网络构架模式。

拓展:

jquery的发展历程:

jQuery 1.0

（2006年8月）：该库的第一个稳定版本，已经具有了对CSS选择符、事件处理和

AJAX交互的稳健支持。

jQuery 1.1

（2007年1月）：这一版大幅简化了API。许多较少使用的方法被合并，减少了需要掌握和解释的方法数量。

jQuery 1.1.3

（2007年7月）：这次小版本变化包含了对jQuery选择符引擎执行速度的显著提升。从这个版本开始，jQuery的性能达到了Prototype、Mootools以及Dojo等同类JavaScript库的水平。

jQuery 1.2

（2007年9月）：这一版去掉了对XPath选择符的支持，原因是相对于CSS语法它已经变得多余了。这一版能够支持对效果的更灵活定制，而且借助新增的命名空间事件，也使插件开发变得更容易。

jQuery UI（2007年9月）：这个新的插件套件是作为曾经流行但已过时的Interface插件的替代项目而发布的。jQuery UI中包含大量预定义好的部件（widget），以及一组用于构建高级元素（例如可拖放的界面元素）的工具。

jQuery 1.2.6

（2008年5月）：这一版主要是将Brandon Aaron开发的流行的Dimensions插件的功能移植到了核心库中。

jQuery 1.3

（2009年1月）：这一版使用了全新的选择符引擎Sizzle，库的性能也因此有了极大提升。这一版正式支持事件委托特性。

jQuery 1.3.2

（2009年2月）：这次小版本升级进一步提升了库的性能，例如改进了：visible/:hidden选择符、.height()/.width()方法的底层处理机制。另外，也支持查询的元素按文档顺序返回。

jQuery 1.4

（2010年1月14号）：对代码库进行了内部重写组织，开始建立一些风格规范。老的core.js文件被分为attribute.js,css.js,data.js,manipulation.js,traversing.js和queue.js；CSS

和attribute的逻辑分离

jQuery 1.5

（2011年1月31日）：该版本修复了83个bug，解决了460个问题。重大改进有：重写了Ajax模块；新增延缓对象（Deferred Objects）；jQuery替身——jQuery.sub()；增强了遍历相邻节点的性能；jQuery开发团队构建系统的改进。

jQuery 1.7

2011年09月29日jQuery 1.7 的第一个 beta 测试版本，该版本修复了超过 50 个的问题以及带来一些新特性。

2011年11月4日jQuery1.7正式版发布

jQuery 1.7.2

2012年03月24日jQuery 1.7.2正式版发布。

该版本在1.7.1的基础上修复了大量的bug，并改进了部分功能。而相比于1.7.2 RC1，只修复了一个bug。值得注意的是：如果你正在使用jQuery Mobile，请使用最新的jQuery 1.7.2和jQuery Mobile 1.1这两个版本，因为之前的jQuery Mobile版本还基于jQuery core 1.7.1或更早的版本。

jQuery 1.8.3

2012年11月14日 jQuery 1.8.3 发布，修复 bug 和性能衰退问题

jQuery 2.0

2013年3月 jQuery 2.0 Beta 2 发布

据jQuery官方博客3月消息，jQuery 2.0 Beta 2 发布。

根据用户对jQuery 2.0 Beta 1 版本的反馈，Beta 2 版做了一些修改。jQuery官方表示，非常需要用户来测试 Beta 2 版，最好同时也能向他们反馈提交建议。

他们相信，Beta 2 版已非常稳定，值得一试，不需要等 2.0 的最终版本。

jQuery 团队在官博中再次提醒用户，jQuery 2.0 不再支持IE 6/7/8 了，但是 jQuery 1.9 会继续支持。因为旧版 IE 浏览器在整个互联网中还有很大部分市场，所以他们非常期望大部分网站能继续使用 jQuery 1.x 一段时间。jQuery 团队也将同时支持 jQuery 1.x 和 2.x 。1.9 和 2.0 版的 API 是相同的，所以不必因为你们网站还在用 jQuery 1.9，就感觉好像错过了什么，或者是落后了。

jQuery 1.9.1汉化版2013年2月23日发布

jQuery2.1.0

2014年1月24日，jQuery2.1.0版发布 [1]

jQuery2.1.1

2014年5月1日，jQuery2.1.1版发布

版本问题（版本号及版本概念）

在开发中用途。

1.改朝换代级别的。 大的版本号变动。不向下兼容。

2.换了一个皇帝。小的版本号更新。 向下兼容。

3.皇帝推行了一个新的政策。微版本更新。

是否每次更新都需要重新学习?

jquery的更新是延续久版本进行的一些更新，是原有内容的延伸并不是彻底的推翻重做，所以更新版本的学习成本非常低。

为什么jquery如此优秀。

1.利用css优势查找页面元素的机制构建于css选择符上。

2.支持宽展（jquery将特殊情况下使用的工具归入插件当中，创建和使用插件的方法非常简单）

3.* 抽象浏览器不一致 （每种浏览器对颁布的标准都有自己的一套不太一致的实现方案，任何web 应用程序都包含有处理这些平台间

特性差异的重要组成部分。jquery 用一个抽象层，来标准化日常任务，从而有效的减少了代码量，同时，也极大的简化了这些任务）

4.*将多重操作集于一行（为了必变过度使用临时变量或不必要的代码重复，jquery在其多数方法中使用了连缀的编程模式）

两种确定是否成功引入jquery库的方法。

1.打开浏览器，soucres资源选项里看， 是否jQuery成功引入。

2.在script代码里查看是否能够调用jquery方法。

引出条件， jquery语法沿用了js的语法，但定制的API会有特殊使用方法，特殊之处在后面的学习之中慢慢发掘。

案例1：

搭建jquery环境。

jQuery核心

window.onload是原生方法， jquery（）是jquery方法， 他们在页面中能不能共存那?

jquery中的window.onload=function(){}方法是
jQuery(document).ready(function(){})

他和window.onload有什么不同那?

实验证明， .ready()方法他的执行速度更快于window.onload 。内部原理忽略不计。

可以共存就意味着可以同时使用。但是并不意味着， 这两个方法同

时存在是合理的。一个页面我们习惯只放一个.ready()方法。 这样不会让页面的逻辑混乱，会增加代码的可读性 。

推荐用.ready()方法，可以排除jquery未引入错误。

习惯， 页面中只用一个.ready()方法（可以存在多个，但是并不科学。）

简写：

```
$(function(){

})
```

拓展2：

项目以及兼容性问题

兼容性问题的两种说法。 兼容或不兼容所有浏览器的代价。

1.成本问题。
2.用户的选择。（高质量用户， 低质量d用户。是否付费|是否有消费能力）
3.项目侧重点。微博不兼容Ie6 ie7
4.用户体验。两种解决方案， 一种高版本 ， 一种低版本。 另一种兼顾低版本，两种都显示低版本。
5.数据统计。
6.教育用户，是否有固定的客户群。抓牢高质量用户。低版本用户产生提示。

jQuery选择器

jQuery最有用的部分：jQuery选择器引擎！jQuery的选择器依赖css1~css3的选择器、。

1.简单选择器：使用频率： *****

1）选择器写法：$( ) >>>>> $('div') 字符串

根据css样式来进行选取。css叫做添加样式，但是jQuery叫做添加行为。

选择器	CSS模式	jQuery模式	描述
标签名	div{}	$('div')	获取所有div标签的DOM元素

ID	#box{}	$('#box')	获取一个ID为box的DOM对象
class(类名)	.box{}	$('.box')	获取所有class名为box的DOM对象

案例3：选择器的使用

jQuery最常见的选择器就是这些。

一个小的知识点：ID选择器的失明现象。

ID在一个页面中只能出现一次，这是一个唯一标识符。在jQuery中就会存在问题。

选择器选择出的对象，有一些属性和方法（length,.size()）

jQuery的兼容方法

css3的子选择器（不兼容IE6）

但是到了jQuery中，jQuery会自行将不兼容IE的问题解决掉。

box>p

1.jQueryDOM对象和原生JavaScriptDOM对象之间的属性方法是否通用。

相互转换：

$('DOM')[0].style.color=red;

$('DOM').get[0].style.color=red;

进阶选择器：

选择器	CSS模式	jQuery模式	描述
群组选择器	div,span,p{}	$('div,span,p')	略
后代选择器	ul li a{}	$('ul li a')	略
通配选择器	*{}	$('*')	略

通配选择器：选择所有；对性能有极大的浪费所以不能在全局范围内使用，最好的方法就是在局部环境下使用；

$('ul li a,ul li em,ul li strong')

简化成通配选择器：

$('ul li *')

高级选择器：

1）层次选择器

选择器	css模式	jQuery模式	描述
后代选择器	ul li a{}	$('ul li a')	获取追溯到的所有元素
子选择器	div>p{}	$('div>p')	只获取子类节点

next选择器	div+p{}	$('div+p')	只获取某节点后一个同级DOM元素
nextAll选择器	div~p{}	$('div~p')	获取某节点后所有同级DOM元素

jQuery为后代选择器提供了一个方法**find()**这个find方法里面有一个参数，就是想要找到的后代（可以是标签，类名，ID）

$('div p').css('color','red') == $('div').**find('p')**.css('color','red')

jQuery为子选择器提供了一个方法，**children()**，参数同上；

$('div>p').css('color','red') ==
$('div').**children('p')**.css('color','red');

jQuery提供了**next()** ，**nextAll( )** 选择器，参数同上：注意next（）选择器，只选择后一个元素。

$('div+p').css('color','red') == $('div').**next('p')**.css('color','red')

$('div~p').css('color','red') == $('div').**nextAll('p')**.css('color','red')

注意：children() , next() , nextAll() 这些方法如果不传递参数的话，那么就默认传递一个通配符*，通常在使用这些选择器的时候，我们需要精准的选择元素，避免发生各种怪异结果。

属性选择器：

CSS模式	jQuery模式	描述

input[name]	$('input[name]')	获取具有这个属性的DOM元素
input[name=XXX]	$('input[name=XXX]')	获取具有属性且属性值为XXX的DOM元素
input[value][name=XXX]	$('input[button][name=XXX]')	获取有value 属性且name为XXX的DOM元素

更多组合等待你来发掘；

过滤选择器：

伪类选择器：

过滤器名	jQuery语法	说明	返回
:first	$('li:first')	选取第一个元素	单个元素
:last	$('li:last')	选取最后一个元素	单个元素
：not(选择器)	$('li:not(.red)')	选取class不是red的元素	一组元素
:even	$('li:even')	选择偶数的所有元素	一组元素

:odd	$('li:odd')	选择所有奇数元素	一组元素
： eq	$('li:eq（1）')	选择对应下表的元素	单个元素

内容过滤器

过滤器名	jQuery语法	说明	返回
:contains(text)	$('li:contains(123456)')	选择有123456文本的元素	一组元素
:empty	$(li':empty')	选取li中不包含子元素或空文本的元素	一组元素
：has（选择器）	$('ul:has(.red)')	选择子元素含有类red的ul	一组元素

jQuery为了优化:has选择器性能，提供了一个方法.has()

$('ul:has(.red)')==$('ul').has('.red')

可见性选择器

过滤器名	jQuery语法	说明	返回
:hidden	$(li:hidden)	选取所有不可见元素	集合元素
:visible	$('li:visible')	选取所有可见元素	集合元素

注：是否可见的判定因素为display：block & display :none