JQuery入门实战

JQuery RUMEN SHIZHAN

主编 汤 东 张富银

JQuery是一个快速、简单的JavaScript library，
它简化了HTML文件的traversing、事件处理、动画、AJAX互动，
从而方便了网页制作的快速发展。

本书详细地讲解了JQury的各种方法和使用技巧，
读者可以系统地掌握JQuery中关于
DOM操作、事件、动画效果、表单操作、Ajax以及插入方面的知识点。
本书适合所有对JQuery技术感兴趣的
WEB设计的前端开发人员、后端开发人员阅读学习。

西南财经大学出版社
Southwestern University of Finance & Economics Press

JQuery入门实战

汤东　张富银　主编

西南财经大学出版社

前言

一、编写本书的目的

市场上JQuery的书籍全部都只写了基础的使用，并没有利用JQuery开发一套完整的demo案例，造成新手入门难、熟手不愿意看的局面。本书打破了传统编写手法，全部采用真实案例讲解，并且保证所有源代码均能正常运行。

二、本书主要讲解的内容

本书详细地讲解了JQuery的各种方法和使用技巧，读者可以系统地掌握JQuery中关于DOM操作、事件、动画效果、表单操作、AJAX以及插入方面的知识点，并且在本书的第十六章我们会参考成熟案例详细讲解JQuery项目的开发，为新手入门打下坚实基础。

本书共分十六章。

第一章首先介绍了什么是JQuery、学习JQuery的条件、JQuery的版本、JQuery的功能和优势、其他JavaScript库、是否兼容低版本IE、下载及运行JQuery。

第二章介绍了JQuery的基础核心内容，包含代码风格、加载模式、对象互换、多个库之间的冲突。

第三章主要讲解常规选择器，本章节是JQuery入门的关键，主要由简单选择器、进阶选择器、高级选择器组成。

第四章主要讲解过滤选择器，包括基本过滤器、内容过滤器、可见性过滤器、子元素过滤器、其他方法。

第五章主要讲解基础DOM和CSS操作，包括DOM简介、设置元素及内容、元素属性操作、元素样式操作、CSS方法。

第六章主要讲解DOM节点操作，包括创建节点、插入节点、包裹节点、节点操作。

第七章主要讲解表单选择器，包括常规选择器、表单选择器、表单过滤器。

第八章主要讲解事件中的基础事件，包括绑定事件、简写事件、复合事件。

第九章主要讲解事件对象中的基础事件对象和冒泡与默认行为。

第十章主要讲解事件中的高级事件部分，包括使用最多的模拟操作、命名空间、事件委托、On、Off和One。

第十一章主要讲解JQuery中的动画效果，包括动画的显示、隐藏、滑动、卷动、淡入、淡出、自定义动画、列队动画方法、动画相关方法、动画全局属性。

第十二章讲解JQuery中的AJAX应用，首先介绍了AJAX的优势与不足，讲解了load（）方法、$.get（）和$.post（）、$.getScript（）和$.getJSON（）、$.ajax（）方法、表单序列化。

第十三章讲解AJAX的进阶应用，主要解决了具体的AJAX使用中最常遇到的问题及解决方法。AJAX加载请求、AJAX错误处理、AJAX请求全局事件、AJAX的跨域JSON和JSONP、jqXHR对象。

第十四章讲解JQuery工具虑数，如字符串操作、数组和对象操作、测试操作、URL操作、浏览器检测、其他操作。

第十五章讲解JQuery的插件机制以及开发自己想要的插件，主要包括插件概述、验证插件、自动完成插件、自定义插件。

第十六章将用前面第一章至第十五章全部知识开发一套完成的项目，项目主要参考知乎网（http://www.zhihu.com）。在项目中将使用到JQuery UI、邮箱自动补全、日历UI、验证插件（validate）、form表单插件、cookie插件、AJAX登录、AJAX提问、AJAX显示问题、AJAX提交评论、AJAX显示评论、AJAX加载更多评论、处理错误与屏蔽低版IE等。

三、本书适合您吗

本书适合所有对JQuery技术感兴趣的WEB设计的前端开发人员、后

端开发人员阅读学习。

阅读此书需要一定的HTML、CSS和JavaScript基础。

四、本书约定

本书所有例子都是基于JQuery1.10版而编写。

如果没有特殊说明，JQuery默认是导入的。

如果没有特殊说明，程序中的$符号都是JQuery的一个简写行为。

如果没有特殊说明，所有网页的头部都必须有标准的DOCTYPE声明。

如果没有特殊说明，所有网页的编码都是UTF-8无BOM格式。

五、读者反馈与示例下载

我们十分欢迎来自读者的宝贵意见与建议，这些建议可以是您感兴趣的内容，或者是本书没有介绍到而又是您十分需要的知识。

作者

2015年5月

目　录

第一章　JQuery入门

教学要点：

1．什么是JQuery

2．学习JQuery的必要条件

3．JQuery的版本

4．JQuery的功能和优势

5．其他Javascript库

6．是否兼容低版本的IE

7．下载及运行JQuery

教学重点：

1．理解框架的概念

2．掌握新技术的学习方法

3．练习JQuery的编程手感

教学难点：

JQuery的编程手感

一、什么是**JQuery**

JQuery是一个Javasript库，通过封装原生的Javascript函数得到一套定义好的方法。JQuery是John Resig于2006年创建的一个开源项目。随着越来越多的开发者加入，JQuery已经集成了Javascript，CSS，DOM，AJAX于一体的强大功能，它可以用更少的代码完成更多更强大更复杂的功能，从而得到开发者的青睐。

二、学习**JQuery**的必要条件

JQuery是Javascript库，所以JQuery在使用上要比Javascript简单，但对于网页编程来说，有些通用的基础知识是必备的：

（1）XHTML或HTML5（含CSS）；

（2）Javascrtpt；

（3）服务器端语言（如PHP、JAVA、.NET）。

三、**JQuery**的版本

2006年8月正式发了JQuery1.0版本，第一个稳定版本，具有对CSS选择符、事件处理以及AJAX的交互支持。

2007年1月发布了JQuery1.1版本，极大地简化了API，合并了许多极少使用的方法。

2007年7月发布了JQuery1.1.3版本，优化了JQuery选择符引擎的执行速度。

2007年9月发布了JQuery1.2版本，优化了Xpath选择器，增加了命名空间事件。

2009年1月发布了JQuery1.3，使用了全新的选择符引擎Sizzle，性能得到进一步提升。

2010年1月发布了JQuery1.4，进行了一次大规模更新，提供了DOM操作，增加了很多新的方法或是增强了原有的方法。

2010年2月发布了JQuery1.4.2，添加了.delegate（）和.undelegate（）两个新方法，提升了灵活性和浏览器的一致性，对事件系统进行了升级。

2011年1月发布了JQuery1.5，重写了AJAX组件，增强了扩展性和性能。

2011年5月发布了JQuery1.6，重写了Attribute组件，引入了新对象

和方法。

2011年11月发布了JQuery1.7，引入了.on（）和.off（）简介的API解决事件绑定及委托容易混淆的问题。

2012年3月发布了JQuery1.7.2，进行一些优化和升级。

2012年7月发布了JQuery1.8，8月发布了JQuery1.8.1，9月发布了JQuery1.8.2，重写了选择符引擎，修复了一些问题。

2013年1月发布了JQuery1.9，CSS的多属性设置，增强了CSS3。

2013年5月发布了JQuery1.10，增加了一些功能。2013年4月发布了JQuery2.0，5月发布了JQuery2.0.2，一个重大更新版本，不再支持IE6/7/8，体积更小、速度更快。

本书我们使用的是最新的中文版的API文档（1.8版本），有在线和离线两种手段查阅：①在线的AP文档可以访问：http://t.mb5u.com/jquery/。②离线的AP文档将打包提供给大家。

其中，版本号升级主要有三种：第一种是大版本升级，比如1．×.×升级到2．×.×，这种升级规模是最大的，改动的地方是最多的，周期也是最长的。比如JQuery从1．×.×到2．×.×用了7年。第二种是小版本更新，比如从1.7升级到1.8，改动适中，增加或减少了一些功能，一般周期半年到一年左右。第三种是微版本更新，比如从1.8.1升级到1.8.2，修复一些bug或错误之类。

版本的内容升级也主要有三种：第一种是核心库的升级，比如优化选择符、优化DOM或者AJAX等；这种升级不影响开发者的使用。第二种是功能性的升级，比如剔除一些过时的方法、新增或增强一些方法等。这种升级需要了解和学习。第三种就是BUG修复之类的升级，对开发者使用没有影响。

学习者有一种担忧，比如学了1.3版本的JQuery，那么以后升级新版本是不是还需要重学？没必要，因为并不是每次升级一个版本都会增加或剔除功能的，一半左右都是内部优化，升级到新版本并不需要任何学习成本。就算在新的版本增加了一些功能，只需要几分钟了解一下即可使用，无需清零之前的知识，只需后续累加。当然，在早期的JQuery版

本都创建了最常用的功能，而新版本中增加的功能，也不是最常用的，无需立即学习，立马用起。

四、**JQuery**的功能和优势

JQuery作为封闭的JavaScript库，其目的就是简化开发者使用JavaSript。主要的功能有以下几点：

（1）像CSS哪样访问和操作DOM；

（2）修改CSS控制页面外观；

（3）简化JavaSript代码操作；

（4）事件处理更加容易；

（5）各种动画效果使用方便；

（6）让AJAX技术更加完美；

（7）基于JQuery的大量插件；

（8）自动扩展功能插件。

JQuery最大的优势就是使用特别方便，比如模仿CSS获取DOM对象，比原生的JS要简单和方便得多，并且在多个CSS的集中处理上非常舒服。而最常用的CSS功能又封装了单独的方法，感觉非常有心。最重要的是JQuery的代码兼容性非常好，你不需要总是去考虑不同浏览器的兼容问题。

五、其他的**Javascript**库

目前除了JQuery外还有五个库比较流行，分别是YUI，Prototype，MooTools，DOJO和ExtJs。

YUI是雅虎公司开发的一套完备的、扩展性良好的富交互网页工具集。

Prototype是最早成型的JavaSctipt库之一，对JavaScript内置对象做了

大量的封装和扩展。

MooTools是一个简洁、模块化的面向对象的JavaScript框架。

DoJo最强大的在于提供其他库没有的功能，如离线存储、图标资源等。

EXtjs简称Ext，原本是对YUI的一个扩展，主要用于创建前端用户界面（收费）。

六、是否兼容**IE**低版本

这次JQuery发布了大版本2．×.×，完全放弃了兼容IE6/7/8。不仅如此，很多国际上的大型站点也开始逐步不再支持IE6/7/8。但对于国内而言，比较大型的网站最多只是抛弃IE6，或者部分功能不支持IE6的警示框，还没可能一下子把IE6/7/8全面抛弃。这里我们就谈一谈你的项目是否有必要兼容IE6/7/8。

完全不支持IE6的示例：网易云课堂——http://study.163.com

完全不支持的做法，就是检测到是否为IE6或IE6、IE8，然后直接跳转到一个信息错误界面，让你更换或升级浏览器，否则无法访问使用。

部分功能不支持的做法，就是判断你是IE6或IE6、IE8，然后给一个警示条或弹出窗，告诉你使用此款浏览器性能降低或部分功能使用不正常或不能使用的提示，但还可以访问使用。

虽然大部分国内网站用IE6去运行都能基本兼容，但很多细节上还是有些问题，导致不能流畅的去使用。

这个问题争论很久，支持兼容的人会拿国情和使用率来证明。不支持兼容的人会用技术落后导致整个水平落后来证明。其实这两种说法都有值得商榷的地方。

首先，传统行业失败率为97％，而新的IT行业的失败率更高达99％以上（数据可能不精确，但可以说明失败率很高）。那么站在更高的角度去看你的项目，你不管是付出3倍成本去完成一个用户体验一般但兼容性很好的项目，还是付出正常成本去完成用户体验很好但不兼容低版

本浏览器。这两种情况不管是哪一种，最终可能都会失败。那么你愿意选择哪种？

是否兼容IE6或IE6、IE8并不单纯是用户基数和国情的问题，而很多项目发起人只一味地用这种理由去判定需求，那么失败也在所难免。除此之外，我们还应该考虑以下几个方面的问题：

1．成本控制

很多项目往往在6、12、18、32个月就会发生财务问题，比如资金紧缩甚至断裂。所以，成本控制尤为重要。项目如果不是老站升级，也不是大门户的新闻站，成本控制和尽快上线测试才是最重要的。而如果新站一味要求全面兼容，会导致成本增加（随着功能多少，成本倍率增加）。为了抓紧时间，就不停地加班再加班，又导致员工产生抵触情绪，工作效率降低，人员流动开始频繁，新员工又要接手开发一半的项目。这样成本不停地在累加。最终不少项目根本没上线就失败了。

2．用户选择

一般可以分为两种用户：高质量用户和低质量用户。所谓高质量用户，就是为了一款最新的3D游戏去升级一块发烧级的显卡，或直接换一台整机。所谓低质量用户，就是发现不能玩最新的3D游戏，就放弃了，去玩“植物大战僵尸”解解馋算了。在用户选择上有一个很好的案例，就是移动互联网。网易和腾讯在它们的新闻应用上，几乎兼容了所有的手机平台，比如IOS、安卓、黑莓、塞班等，因为新闻应用的核心在新闻，而新闻的用户基数巨大，需要兼顾高质量用户和低质量用户。而腾讯在IOS上的几十个应用，除了新闻、QQ、浏览器，其他的基本都只有IOS和安卓，在塞班和黑莓及其他应用上就没有了。

所以，你的应用核心是哪方面？兼容的成本有多大？会不会导致成本控制问题？用户选择尤为重要，放弃低质量用户也是一种成本控制。在用户基数庞大的项目上，放弃低质量用户就有点愚笨，比如某个新闻站有1亿用户，2000万为使用低版本浏览器的低质量用户，而面对2000万用户，你兼容它或单独为2000万用户做个低版本服务，成本虽然可能还是3倍，但从庞大的用户基数来看，这种成本又非常低廉。而你的用户基数只有1000人，而低质量用户有50人，那么为了这50人去做兼容，那么3倍的成本就变得非常昂贵。

3．项目的侧重点

你的项目重点在哪里？是为了看新闻？是为了宣传线下产品？那么你其实有必要兼容低版本浏览器。首先这种类型的站不需要太好的用户体验，不需要太多的交互操作，只是看，而兼容的成本比较低，并且核心在新闻或产品！但如果你的项目有大量的交互、大量的操作，比如全球最大的社交网已经不兼容IE6、IE7，最大的微博也不再兼容IE6、IE7，就是这个原因。所以，项目并不是一味地全面兼容，或者全面不兼容，主要看你的项目侧重点在哪里！

4．用户体验

如果你的项目在兼容低版本浏览器成本巨大，比如社交网，有大量的JS和AJAX操作。那么兼容IE6、IE7的成本确实很高，如果兼容，用户体验就会很差。兼容有两种：一种是高版本浏览器用性能好，体验好的模式；低版本的自动切换到兼容模式。另一种就是，不管高版本或低版本都用统一的兼容模式。这两种成本都很高。用户体验好的模式，能增加用户黏度，增加付费潜在用户，而用户体验差的总是被用户归纳为心目中的备胎（所谓备胎就是实在没有了才去访问，如果有，很容易被抛弃）。

5．数据支持

如果对某一种类型的网站项目有一定的研究，那么手头必须有支持的数据分析。有数据分析可以更好地进行成本控制，更有魄力地解决高质量用户和低质量用户的取舍。

6．教育用户

很多项目可能是有固定客户群，或者使用该项目人员质量普遍较高。那么，面对零星一点的低质量用户，我们不能再去迎合他。因为迎合他，就无法用高质量用户体验去粘住忠实用户，同时也不能获取低质量用户的芳心。所以，我们应有的策略是：牢牢把握住高质量的忠诚用户，做到他们心目中的第一；教育那部分低质量用户（比如企业级开发项目，可以直接做企业培训，安装高版本浏览器等。互联网项目，就给出提示安装高版本浏览器即可）。那么一部分低质量用户被拉拢过来，还有一小撮死性不改的用户就只有放弃。切不可捡了芝麻丢了西瓜，不要贪大求全。

7．经验之谈

以上我们讨论了是否需要兼容IE6或IE7、IE8，结论就是必须根据实际情况，即成本情况、人员情况、用户情况和项目本身类型情况来制定，没有一刀切的兼容或不兼容。

七、下载及运行**JQuery**

目前最新的版本，是JQuery1.10.1和JQuery2.0.2，我们下载开发版，可以顺便读一读源代码。如果你需要引用到你线上的项目，就必须使用压缩版，去掉注释和空白，使容量最小。

本课程使用的软件是：Nodepad++使用测试的浏览器为：Firefox3.6.8、Firefox21+、Chrome、IE6/7/8/9、Opera和Safari。

使用的版本为：JQuery1.10.1和JQuery2.02。

使用的html版本为：xhtml1.0，在必要的时候将会使用html5。

使用的调试工具为：Firefox下的firebug。

测试代码

```
//单击按钮弹窗
$（function （） {
$（'input'）.click（function （） {
   alert（'第一个JQuery程序! '）;
}）;
}）;
```

第二章　基础核心

教学要点：

1．代码风格；

2．加载模式；

3．对象互换；

4．多个库之间的冲突。

教学重点：

1．熟悉JQuery的代码风格；

2．了解JQuery的加载模式；

3．对象互换；

4．多个库之间的冲突。

教学难点：

对象互换、多个库之间的冲突。

开篇：本节课我们简单地介绍一下JQuery一些核心的问题，这些问题为后续课程展开提供了帮助。对于JavaScript课程已经学完的同学，这些概念会非常清晰，而对于JavaScript薄弱的同学可能会有一些模糊，但不必太担心，后续会慢慢展开。而对于完全没有JavaScript基础的同学，就无法学习了。

一、**JQuery**代码风格

在JQuery程序中，不管是页面元素的选择还是内置的功能函数，都是从美元符号“$”来起始的。而这个“$”就是JQuery当中最重要且独有的对象，所以我们在页面元素选择或执行功能函数的时候可以这么写：

```
$（function （） {}）； //执行一个匿名函数
$（‘#box’）； //进行执行的ID元素选择
$（‘#box’）.css（'color'，'red'）； //执行功能函数
```

由于$本身就是JQuery对象的缩写形式，那么也就是说上面的三段代码也可以写成如下形式：

```
JQuery（function （） {}）；
JQuery（'#box'）；
JQuery（'#box'）.css（'color'，'red'）。
```

在执行功能函数的时候，我们发现.css（）这个功能函数并不是直接被“$”或JQuery对象调用执行的，而是先获取元素后，返回某个对象再调用.css（）这个功能函数。那么也就是说，这个返回的对象其实也就是JQuery对象。

$（）.css（'color'，'red'）；//理论上合法，但实际上缺少元素而报错。

值得一提的是，执行了.css（）这个功能函数后，最终返回的还是JQuery对象，那么也就是说，JQuery的代码模式是采用的连缀方式，可以不停地连续调用功能函数。

$（'#box'）.css（'color'，'red'）.css（'font-size'，'50px'）；//连缀

JQuery中代码注释和JavaScript是保持一致的，有两种最常用的注释：单行使用“//...”；多行使用“/*...*/”。//$（'#box'）.css（'color'，'red'）。

二、加载模式

我们在之前的代码一直在使用$（function（）{}）；这段代码进行首尾包裹，那么为什么必须要包裹这段代码呢？原因是JQuery库文件是在body元素之前加载的，我们必须等待所有的DOM元素加载后，延迟支持DOM操作，否则就无法获取到。在延迟等待加载，JavaScript提供了一个事件为load。其方法如下：

```
window.onload = function （） {}； //JavaScript等待加载。
$（document）.ready（function （） {}）； //JQuery等待加载。
```

表2-1 onload和ready的区别

	window.onload	$（document）.ready（）
执行时机	必须等待网页全部加载完毕（包括图片等），然后再执行包裹代码。	只需要等待网页中的DOM结构加载完毕，就能执行包裹的代码。
简写方案	无	$（function（）{}）；

在实际应用中，我们都很少直接去使用window.onload，因为它需要等待图片之类的大型元素加载完毕后才能执行JS代码。所以，最头疼的就是在网速较慢的情况下，页面已经全面展开，图片还在缓慢加载，这时页面上的JS交互功能全部处在假死状态。

三、对象互换

JQuery对象虽然是JQuery库独有的对象，但它也是通过JavaScript进行封装而来的。我们可以直接输出来得到它的信息。

```
alert（$）； //JQuery对象方法内部。
alert（$（））； //JQuery对象返回的对象还是JQuery。
alert（$（'#box'））； //包裹ID 元素返回的对象还是JQuery。
```

从上面三组代码我们发现：只要使用了包裹后，最终返回的都是JQuery对象。这样的好处显而易见，就是可以连缀处理。但有时我们也需要返回原生的DOM对象，比如：

```
alert（document.getElementById（'box'））； //［object HTMLDivE
JQuery想要达到获取原生的DOM 对象，可以这么处理：
alert（$（'#box'）.get（0））； //ID元素的第一个原生DOM
```

从上面get（0）可以看出，JQuery是可以进行批量处理DOM的，这样可以在很多需要循环遍历的处理上更加得心应手。

四、多个库之间冲突的解决

当一个项目中引入多个第三方库的时候，由于没有命名空间的约束（命名空间就好比同一个目录下的文件夹一样，名字相同就会产生冲突），库与库之间发生冲突在所难免。

那么，既然有冲突的问题，为什么要使用多个库呢？原因是JQuery只不过是DOM操作为主的库，方便我们日常Web开发。但有时我们的项目有更多特殊的功能需要引入其他的库，比如用户界面UI方面的库，游戏引擎方面的库等一系列。

而很多库，比如prototype和Base库，都使用“$”作为基准起始符，如果想和JQuery共容有两种方法：

（1）将JQuery库在Base库之前引入，那么“$”的所有权就归Base库所有，而JQuery可以直接用JQuery对象调用，或者创建一个“$$”符给JQuery使用。

```
var $$ = JQuery;  //创建一个$$的JQuery对象
$(function () { //这是Base 的$
    alert($('#box').ge(0));  //这是Base 的$
    alert($$('#box').width());  //这是JQuery的$$
});
```

（2）如果将JQuery库在Base库之后引入，那么“$”的所有权就归JQuery库所有，而Base库将会冲突而失去作用。这里，JQuery提供了一个方法：

```
JQuery.noConflict();  //将$符所有权剔除
var $$ = JQuery;
$(function () {
    alert($('#box').ge(0));
    alert($$('#box').width());
});
```

第三章　常规选择器

教学要点：

1．简单选择器；

2．进阶选择器；

3．高级选择器。

教学重点：

1．简单选择器；

2．进阶选择器；

3．高级选择器。

教学难点：

理解什么是选择器，多种选择器组合使用。

开篇：JQuery最核心的组成部分就是选择器引擎。它继承了CSS的语法，可以对DOM元素的标签名、属性名、状态等进行快速、准确的选择，并且不必担心浏览器的兼容性。JQuery选择器除实现了CSS1~CSS3的大部分规则之外，还实现了一些自定义的选择器，用于各种特殊状态的选择。备注：学习本课程必须有（X）html+CSS基础。

一、简单选择器

在使用JQuery选择器时，我们首先必须使用“$（）”函数来包装我们的CSS规则。而CSS规则作为参数传递到JQuery对象内部后，再返回包含页面中对应元素的JQuery对象。随后，我们就可以对这个获取到的DOM节点进行行为操作了。

```
#box { //使用ID 选择器的CSS 规则：
    color：red；  //将ID 为box 的元素字体颜色变红
```

}

在JQuery选择器里，我们使用如下方式来获取同样的结果：

$（'#box'）.css（'color'，'red'）；//获取DOM节点对象，并添加行为。

那么除了ID选择器之外，还有两种基本的选择器：元素标签名和类（class）。

表3-1

选择器	CSS模式	JQuery模式	描述
元素标签名	Div{}	$（'div'）	获取所有div元素的DOM对象
ID	#box {}	$（'#box'）	获取一个ID为box元素的DOM对象
类（class）	.box{}	$（'.box'）	获取所有class为box的所有DOM对象

$（'div'）.css（'color'，'red'）；//元素选择器，返回多个元素。

$（'#box'）.css（'color'，'red'）；//ID选择器，返回单个元素。

$（'.box'）.css（'color'，'red'）；//类（class）选择器，返回多个元素。

为了证明ID返回的是单个元素，而元素标签名和类（class）返回的是多个，我们可以采用JQuery核心自带的一个属性length或size（）方法来查看返回的元素个数。

alert（$（'div'）.size（））；//3个。

alert（$（'#box'）.size（））；//1个，后面两个失明了。

alert（$（'.box'）.size（））；//3个。

同理，你也可以直接使用JQuery核心属性来操作：

alert（$（'#box'）.length）；//1个，后面失明了。

警告：有个问题特别要注意，ID在页面只允许出现一次，我们一般都是要求开发者要遵守和保持这个规则。但如果你在页面中出现三次，并且在CSS使用样式，那么这三个元素还会执行效果。但如果你想要JQuery这么去做，那么就会遇到失明的问题。所以，开发者必须养成良好的遵守习惯，在一个页面仅使用一个ID。

$（'#box'）.css（'color'，'red'）；//只有第一个ID变红，后面两个ID失明了。

JQuery选择器的写法与CSS选择器十分类似，只不过它们的功能不同。CSS找到元素后添加的是单一的样式，而JQuery则添加的是动作行为。最重要的一点是：CSS在添加样式的时候，高级选择器会对部分浏览器不兼容，而JQuery选择器在添加CSS样式的时候却不必为此烦恼。

```
#box > p {//CSS 子选择器，IE6不支持
    color：red；
}
$（'#box > p'）.css（'color'，'red'）；  //JQuery子选择器，兼容了IE
```

JQuery选择器支持CSS1、CSS2的全部规则，支持CSS3部分实用的规则，同时它还有少量独有的规则。所以，对于已经掌握CSS的开发人员，学习JQuery选择器几乎是零成本。而JQuery选择器在获取节点对象的时候不但简单，还内置了容错功能，这样避免像JavaScript那样每次对节点的获取需要进行有效判断。

$（'#pox'）.css（'color'，'red'）；//不存在ID为pox的元素，也不报错。

document.getElementById（'pox'）.style.color = 'red'；//报错了。

因为JQuery内部进行了判断，而原生的DOM节点获取方法并没有进行判断，所以导致了一个错误，原生方法可以这么判断解决这个问题：

```
if （document.getElementById（'pox'）） { //先判断是否存在这个对象
   document.getElementById（'pox'）.style.color = 'red'；
}
```

那么对于缺失不存在的元素，我们使用JQuery调用的话，怎么去判断是否存在呢？因为本身返回的是JQuery对象，可能会导致不存在元素存在与否，都会返回true。

```
if （$（'#pox'）.length > 0） { //判断元素包含数量即可
   $（'#pox'）.css（'color'， 'red'）；
}
```

除了这种方式之外，还可以用转换为DOM对象的方式来判断，例如：

if（$（'#pox'）.get（0））{}或if（$（'#pox'）［0］）{} //通过数组下标也可以获取DOM对象。

二、进阶选择器

在简单选择器中，我们了解了最基本的三种选择器：元素标签名、ID和类（class）。那么在基础选择器外，还有一些进阶和高级的选择器方便我们进行更精准的选择元素。

表3-2

选择器	CSS模式	JQuery模式	描述
群组选择器	span， em，.box {}	$（'span， em，.box'）	获取多个选择器的DOM对象
后代选择器	ul li a {}	$（'ul li a'）	获取追溯到的多个DOM对象
通配选择器	* {}	$（'*'）	获取所有元素标签名的DOM对象

```
//群组选择器
```

```
span， em， .box { //多种选择器添加红色字体
    color：red；
}
$（'span， em， .box'）.css（'color'， 'red'）； //群组选择器JQue
//后代选择器
ul li a { //层层追溯到的元素添加红色字体
    color：red；
}
$（'ul li a'）.css（'color'， 'red'）； //群组选择器JQuery方式
//通配选择器
* { //页面所有元素都添加红色字体
    color：red；
}
$（'*'）.css（'color'， 'red'）； //通配选择器
```

目前介绍的六种选择器，在实际应用中，我们可以灵活地搭配，使得选择器更加精准和快速：

$（'#box p，ul li *'）.css（'color'，'red'）；//组合了多种选择器。

警告：在实际使用上，通配选择器一般用得并不多，尤其是在大通配上，比如：$（'*'）。这种使用方法效率很低，影响性能，建议尽可能少用。还有一种选择器，可以在ID和类（class）中指明元素前缀，比如：

$（'div.box'）；//限定必须是.box元素

$（'p#box div.side'）；//同上

类（class）有一个特殊的模式，就是同一个DOM节点可以声明多个类（class）。那么对于这种格式，我们有多class选择器可以使用，但要注意和class群组选择器的区别。

```
.box.pox { //双class 选择器，IE6 出现异常。
    color：red；
}
$（'.box.pox'）.css（'color'， 'red'）； //兼容IE6，解决了异常。
```

多class选择器是指必须一个DOM节点同时有多个class，用这些class进行精确限定。而群组class选择器，只不过是多个class进行选择而已。

$（'.box，.pox'）.css（'color'，'red'）；//加了逗号，体会区别。

警告：在构造选择器时，有一个通用的优化原则：只追求必要的确定性。当选择器筛选越复杂，JQuery内部的选择器引擎处理字符串的时间就越长。比如：

$（'div#box ul li a#link'）；//让JQuery内部处理了不必要的字符串

$（'#link'）；//ID是唯一性的，准确度不变，性能提升

三、高级选择器

在前面我们学习了六种最常规的选择器，一般来说通过这六种选择器基本上可以解决所有DOM节点对象选择的问题。但在很多特殊的元素上，比如父子关系的元素、兄弟关系的元素、特殊属性的元素等。在早期CSS的使用上，由于IE6等低版本浏览器不支持，所以这些高级选择器的使用也不具备普遍性，但随着JQuery兼容，这些选择器的使用频率也越来越高。

表3-3　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'）	获取追溯到的多个DOM对象
子选择器	div > p {}	$（'div p'）	只获取子类节点的多个DOM对象
next选择器	div + p {}	$（'div + p'）	只获取某节点后一个同级DOM对象
nextAll选择	div ~ p {}	$（'div ~ p'）	获取某节点后面所有同级DOM对象

在层次选择器中，除了后代选择器之外，其他三种高级选择器是不支持IE6的，而JQuery却是兼容IE6的。

//后代选择器

```
$（'#box p'）.css（'color', 'red'）; //全兼容
JQuery为后代选择器提供了一个等价find（）方法
$（'#box'）.find（'p'）.css（'color', 'red'）; //和后代选择器等价
//子选择器，孙子后失明
#box > p { //IE6 不支持
    color：red；
}
$（'#box > p'）.css（'color', 'red'）; //兼容IE6
```

JQuery为子选择器提供了一个等价children（）方法：

```
$（'#box'）.children（'p'）.css（'color', 'red'）; //和子选择器等价
//next选择器（下一个同级节点）
#box + p { //IE6不支持
    color：red；
}
$（'#box+p'）.css（'color', 'red'）; //兼容IE6
JQuery为next 选择器提供了一个等价的方法next（）：
$（'#box'）.next（'p'）.css（'color', 'red'）; //和next 选择器等价
//nextAll选择器（后面所有同级节点）
#box ~ p { //IE6不支持
    color：red；
}
$（'#box ~ p'）.css（'color', 'red'）; //兼容IE6
```

JQuery为nextAll选择器提供了一个等价的方法nextAll（）：

$（'#box'）.nextAll（'p'）.css（'color'，'red'）；//和nextAll选择器等价

层次选择器对节点的层次都是有要求的。比如子选择器，只有子节点才可以被选择到，孙子节点和重孙子节点都无法选择到。next和nextAll选择器必须是同一个层次的后一个和后N个，不在同一个层次就无法选取到了。

在find（）、next（）、nextAll（）和children（）这四个方法中，如果不传递参数，就相当于传递了“*”，即任何节点。我们不建议这么做，不但影响性能，而且由于精准度不佳可能在复杂的HTML结构时产生怪异的结果。

$（'#box'）.next（）；//相当于$（'#box'）.next（'*'）；

为了补充高级选择器的这三种模式，JQuery还提供了更加丰富的方

法来选择元素：

$（'#box'）.prev（'p'）.css（'color'，'red'）；//同级上一个元素

$（'#box'）.prevAll（'p'）.css（'color'，'red'）；//同级所有上面的元素

nextUntil（）和prevUnitl（）方法是选定同级的下面或上面的所有节点，选定非指定的所有元素，一旦遇到指定的元素就停止选定。

$（'#box'）.prevUntil（'p'）.css（'color'，'red'）；//同级上非指定元素选定，遇到则停止

$（'#box'）.nextUntil（'p'）.css（'color'，'red'）；//同级下非指定元素选定，遇到则停止

siblings（）方法正好集成了prevAll（）和nextAll（）两个功能的效果，以及上下相邻的所有元素进行选定：

```
$（'#box'）.siblings（'p'）.css（'color'， 'red'）； //同级上下所有
//等价于下面：
$（'#box'）.prevAll（'p'）.css（'color'， 'red'）； //同级上所有元
$（'#box'）.nextAll（'p'）.css（'color'， 'red'）； //同级下所有元
```

警告：切不可写成“$（'#box'）.prevAll（'p'）.nextAll（'p'）.css（'color'，'red'）；”这种形式，因为prevAll（'p'）返回的是已经上方所有指定元素，然后在nextAll（'p'）选定下方所有指定元素，这样必然出现错误。

从理论上来讲，JQuery提供的方法find（）、next（）、nextAll（）和children（）运行速度要快于使用高级选择器。因为它们实现的算法有所不同，高级选择器是通过解析字符串来获取节点对象，而JQuery提供的方法一般都是单个选择器，是可以直接获取的。但这种快慢的差异，对于客户端脚本来说没有太大的实用性，并且速度的差异还要取决于浏览器和选择的元素内容。比如，在IE6、IE7不支持querySelectorAll（）方法，则会使用“Sizzle”引擎，速度就会慢，而其他浏览器则会很快。有兴趣的可以了解这个方法和这个引擎。

选择器快慢分析：

//这条最快，会使用原生的getElementById、ByName、ByTagName和querySelectorAll（）

$（'#box'）.find（'p'）；

//JQuery会自动把这条语句转成$（'#box'）.find（'p'），这会导致一定的性能损失。它比最快的形式慢了5%~10%。

$（'p'，'#box'）；

//这条语句在JQuery内部，会使用$.sibling（）和javascript的nextSibling（）方法，一个个遍历节点。它比最快的形式大约慢了50%。

$（'#box'）.children（'p'）；

//JQuery内部使用Sizzle引擎，处理各种选择器。Sizzle引擎的选择顺序是从右到左，所以这条语句是先选p，然后再一个个过滤出父元素#box，这导致它比最快的形式大约慢了70%。

$（'#box > p'）；

//这条语句与上一条是同样的情况。但是，上一条只选择直接的子元素，这一条可以选择多级子元素，所以它的速度更慢，大概比最快的形式慢了77%。

$（'#box p'）；

//JQuery内部会将这条语句转成$（'#box'）.find（'p'），比最快的形式慢了23%。

$（'p'，$（'#parent'））；

综上所述，最快的是find（）方法，最慢的是$（'#box p'）这种高级选择器。如果一开始将$（'#box'）进行赋值，那么JQuery就对其变量进行缓存，那么速度会进一步提高。

var box = $（'#box'）；

var p = box.find（'p'）；

注意：我们应该推荐使用哪种方案呢？其实，使用哪种方案都差不多。这里，我们推荐使用JQuery提供的方法。因为不但方法的速度比高级选择器运行得更快，并且它的灵活性和扩展性要高于高级选择器。使用“+”或“~”从字面上没有next和nextAll更加语义化，更加清晰，JQuery的方法更加丰富，提供了相对的prev和prevAll。毕竟JQuery是编程语言，需要能够灵活的拆分和组合选择器，而使用CSS模式过于死板。所以，如果JQuery提供了独立的方法来代替某些选择器的功能，我们还是推荐优先使用独立的方法。

表3-4　属性选择器

CSS模式	JQuery模式	描述
a［title］	$（'a［title］'）	获取具有这个属性的DOM对象。
a［title=num1］	$（'a［title=num1］'）	获取具有这个属性=这个属性值的DOM对象。
a［title^=num］	$（'a［title^=num］'）	获取具有这个属性且与开头属性值匹配的DOM对象。
a［title｜=num］	$（'a［title｜=num］'）	获取具有这个属性且等于属性值或开头属性值匹配后面跟一个“-”号的DOM对象。
a［title$=num］	$（'a［title$=num］'）	获取具有这个属性且与结尾属性值匹配的DOM对象。
a［title!=num］	$（'a［title!=num］'）	获取具有这个属性且不等于属性值的DOM对象。

a［title~=num］	$（'a［title~=num］'）	获取具有这个属性且属性值是以一个空格分割的列表，其中包含属性值的DOM对象。
a［title*=num］	$（'a［title*=num］'）	获取具有这个属性且属性值含有一个指定字串的DOM对象。
a［bbb］［title=num1］	$（'a［bbb］［title=num1］'）	获取具有这个属性且与属性值匹配的DOM对象。

属性选择器也不支持IE6，所以在CSS界面中如果要兼容低版本，那么也是非主流的。但JQuery不必考虑这个问题。

```
//选定这个属性的
a［title］ { //IE6不支持
    color：red；
}
$（'a［title］'）.css（'color'， 'red'）； //兼容IE6 了
//选定具有这个属性=这个属性值的
a［title=num1］ { //IE6不支持
    color：red；
}
    $（'a［title=num1］'）.css（'color'， 'red'）； //兼容IE6 了
//选定具有这个属性且与开头属性值匹配的
a［title∧=num］ { //IE6不支持
    color：red；
}
$（'a［title=∧num］'）.css（'color'， 'red'）； //兼容IE6 了
//选定具有这个属性且等于属性值或开头属性值匹配后面跟一个“-”号
a［title|=num］ { //IE6不支持
    color：red；
}
$（'a［title|="num"］'）.css（'color'， 'red'）； //兼容IE6 了
//选定具有这个属性且与结尾属性值匹配的
a［title$=num］ { //IE6不支持
    color：red；
}
    $（'a［title$=num］'）.css（'color'，'red'）； //兼容IE6 了
//选定具有这个属性且属性值不相等的
a［title!=num1］ { //不支持此CSS 选择器
    color：red；
```

```
}
$（'a［title!=num1］'）.css（'color'，'red'）；  //JQuery支持这种写
//选定具有这个属性且属性值是以一个空格分割的列表，其中包含属性值的
a［title~=num］ { //IE6不支持
    color：red；
}
$（'a［title~=num1］'）.css（'color'，'red'）；  //兼容IE6
//选定具有这个属性且属性值含有一个指定字串的
a［title*=num］ { //IE6不支持
    color：red；
}
$（'a［title*=num］'）.css（'color'，'red'）；  //兼容IE6
//选定具有多个属性且属性值匹配成功的
a［bbb］［title=num1］ { //IE6 不支持
    color：red；
}
$（'a［bbb］［title=num1］'）.css（'color'，'red'）；  //兼容IE6
```

第四章　过滤选择器

教学要点：

1．基本过滤器；

2．内容过滤器；

3．可见性过滤器；

4．子元素过滤器；

5．其他方法。

教学重点：

1．基本过滤器；

2．内容过滤器；

3．可见性过滤器；

4．子元素过滤器。

教学难点：

理解什么是过滤器，多种过滤器组合使用。

开篇：过滤选择器简称过滤器。它其实也是一种选择器，而这种选择器类似于CSS3（http://t.mb5u.com/css3/）里的伪类，可以让不支持CSS3的低版本浏览器也能支持。和常规选择器一样，JQuery为了更方便开发者使用，提供了很多独有的过滤器。

一、基本过滤器

过滤器主要通过特定的过滤规则来筛选所需的DOM元素，和CSS中的

伪类的语法类似：使用冒号（：）开头。

表4-1

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（selector）	$（'li：not（.red）'）	选取class不是red的li元素。	集合元素
：even	$（'li.even'）	选择索引（0开始）是偶数的所有元素。	集合元素
：odd	$（'li：odd'）	选择索引（0开始）是奇数的所有元素。	集合元素
：eq（index）	$（'li：eq（2）'）	选择索引（0开始）等于index的元素。	单个元素
：gt（index）	$（'li：gt（2）'）	选择索引（0开始）大于index的元素。	集合元素
：lt（index）	$（'li.lt（2）'）	选择索引（0开始）小于index的元素	集合元素。
：header	$（':header'）	选择标题元素，h1 ~ h6。	集合元素
：animated	$（':animated'）	选择正在执行动画的元素。	集合元素
：focus	$（'：focus'）	选择当前被焦点的元素。	集合元素

$（'li：first'）.css（'background'，'#ccc'）；//第一个元素

$（'li：last）.css（'background'，'#ccc'）；//最后一个元素

$（'li：not（.red））.css（'background'，'#ccc'）； //非class 为red的元素

$（'li：even'）.css（'background'，'#ccc'）；//索引为偶数的元素

$（'li：odd）.css（'background'，'#ccc'）；//索引为奇数的元素

$（'li：eq（2））.css（'background'，'#ccc'）；//指定索引值的元素

$（'li：gt（2）'）.css（'background'，'#ccc'）；//大于索引值的元素

$（'li：lt（2）'）.css（'background'，'#ccc'）；//小于索引值的元素

$（'：header'）.css（'background'，'#ccc'）；//页面所有h1 ~ h6 元素

注意：：focus过滤器，必须是网页初始状态的已经被激活焦点的元素才能实现元素获取。而不是鼠标点击或者Tab键盘敲击激活的。

$（'input'）.get（0）.focus（）；//先初始化激活一个元素焦点

$（'：focus'）.css（'background'，'red'）；//被焦点的元素

JQuery为最常用的过滤器提供了专用的方法，已达到提到性能和效率的作用：

$（'li'）.eq（2）.css（'background'，'#ccc'）；//元素li的第三个元素，负数从后开始

$（'li'）.first（）.css（'background'，'#ccc'）；//元素li的第一个元素

$（'li'）.last（）.css（'background'，'#ccc'）；//元素li的最后一个元素

$（'li'）.not（'.red'）.css（'background'，'#ccc'）；//元素li不含class为red的元素

注意：：first、：last和first（）、last（）这两组过滤器和方法在出现相同元素的时候，first会实现第一个父元素的第一个子元素，last会实现最后一个父元素的最后一个子元素。所以，如果需要明确是哪个父元素，需要指明：

```
$（'#box li：last'）.css（'background'， '#ccc'）；//#box 元素的
//或
$（'#box li）.last（）.css（'background'， '#ccc'）；//同上
```

二、内容过滤器

内容过滤器的过滤规则主要包含在子元素或文本内容上。

表4-2

过滤器名	JQuery语法	说明	返回
：contains（text）	$（'：contains（"ycku.com"）'）	选取含有"ycku.com"文本的元素。	元素集合
：empty	$（'：empty'）	选取不包含子元素或空文本的元素。	元素集合
：has（selector）	$（'：has（.red）'）	选取含有class是red的元素。	元素集合
：parent	$（'：parent'）	选取含有子元素或文本的元素。	元素集合

//选择元素文本节点含有ycku.com文本的元素。

$（'div：contains（"ycku.com"）'）.css（'background'，'#ccc'）；

$（'div：empty'）.css（'background'，'#ccc'）；//选择空元素

$（'ul：has（.red）'）.css（'background'，'#ccc'）；//选择子元素含有class是red的元素

$（'：parent'）.css（'background'，'#ccc'）；//选择非空元素。

JQuery提供了一个has（）方法来提高：has过滤器的性能：

$（'ul'）.has（'.red'）.css（'background'，'#ccc'）；//选择子元素含有class是red的元素

JQuery提供了一个名称和：parent相似的方法，但这个方法并不是选取含有子元素或文本的元素，而是获取当前元素的父元素，返回的是元素集合。

$（'li'）.parent（）.css（'background'，'#ccc'）；//选择当前元素的父元素

$（'li'）.parents（）.css（'background'，'#ccc'）；//选择当前元素的父元素及祖先元素

$（'li'）.parentsUntil（'div'）.css（'background'，'#ccc'）；//选择当前元素遇到div父元素停止

三、可见性过滤器

可见性过滤器根据元素的可见性和不可见性来选择相应的元素。

表4-3

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

$（'p：hidden）.size（）；//元素p隐藏的元素

$（'p：visible'）.size（）；//元素p显示的元素

注意：hidden过滤器一般包含的内容为：CSS样式为display：none、input表单类型为type="hidden"和visibility：hidden的元素。

四、子元素过滤器

子元素过滤器的过滤规则是通过父元素和子元素的关系来获取相应的元素。

表4-4

过滤器名	JQuery语法	说明	返回
：first-child	$（'li：first-child'）	获取每个父元素的第一个子元素。	集合元素
：last-child	$（'li：last-child'）	获取每个父元素的最后一个子元素。	集合元素
：only-child	$（'li：only-child'）	获取只有一个子元素的元素。	集合元素
：nth-child（odd/even/eq（index））	$（'li：nth-child（even）'）	获取每个自定义子元素的元素（索引值从1开始计算）。	集合元素

$（'li：first-child'）.css（'background'，'#ccc'）；//每个父元素第一个li元素

$（'li：last-child'）.css（'background'，'#ccc'）；//每个父元素最后

一个li元素

$（'li：only-child'）.css（'background'，'#ccc'）；//每个父元素只有一个li元素

$（'li：nth-child（odd）'）.css（'background'，'#ccc'）；//每个父元素奇数li元素

$（'li：nth-child（even）'）.css（'background'，'#ccc'）；//每个父元素偶数li元素

$（'li：nth-child（2）'）.css（'background'，'#ccc'）；//每个父元素第三个li元素

五、其他方法

JQuery在选择器和过滤器上，还提供了一些常用的方法，方便我们开发时灵活使用。

表4-5

方法名	JQuery语法	说明	返回
is（s/o/e/f）	$（'li'）.is（'.red'）	传递选择器、DOM、jquery对象或是函数来匹配元素结合。	集合元素
hasClass（class）	$（'li'）.eq（2）.hasClass（'red'）	其实就是is（"." + class）。	集合元素
slice（start，end）	$（'li'）.slice（0，2）	选择从start到end位置的元素，如果是负数，则从后开始。	集合元素

filter（s/o/e/f）	$（'li'）.filter（'.red'）		集合元素
end（）	$（'div'）.find（'p'）.end（）	获取当前元素前一次状态。	集合元素
contents（）	$（'div'）.contents（）	获取某元素下面所有元素节点，包括文本节点。如果是iframe，则可以查找文本内容。	集合元素

$（'.red'）.is（'li'）；//true，选择器，检测class是否为red

$（'.red'）.is（$（'li'））；//true，JQuery对象集合，同上

$（'.red'）.is（$（'li'）.eq（2））；//true，JQuery对象单个，同上

$（'.red'）.is（$（'li'）.get（2））；//true，DOM对象，同上

$（'.red'）.is（function（）{//true，方法，同上

return $（this）.attr（'title'）== '列表3'；//可以自定义各种判断

}））；

$（'li'）.eq（2）.hasClass（'red'）；//和is一样，只不过只能传递class

$（'li'）.slice（0，2）.css（'color'，'red'）；//前三个变成红色

注意：这个参数有多种传法和JavaScript的slice方法是一样的。比如：slice（2），从第三个开始到最后选定；slice（2，4），第三个和第四个被选定；slice（0，-2），从倒数第三个位置向前选定所有；slice（2，-2），前两个和末尾两个未选定。

```
$（"div"）.find（"p"）.end（）.get（0）；//返回div 的原生DOM
$（'div'）.contents（）.size（）；//返回子节点（包括文本）数量
$（'li'）.filter（'.red'）.css（'background'，'#ccc'）；//选择li
$（'li'）.filter（'.red，：first，：last'）.css（'background'，'#
//特殊要求函数返回
$（'li'）.filter（function （） {
    return $（this）.attr（'class'） == 'red' && $（this）.attr
}）.css（'background'， '#ccc'）；
```

第五章　基础DOM和CSS操作

教学要点：

1．DOM简介；

2．设置元素及内容；

3．元素属性操作；

4．元素样式操作；

5．CSS方法。

教学重点：

1．DOM简介；

2．设置元素及内容；

3．元素样式操作；

4．CSS方法。

教学难点：

理解什么是DOM及怎样去操作DOM。

开篇：DOM是一种文档对象模型，方便开发者对HTML结构元素内容进行展示和修改。在JavaScript中，DOM不但内容庞大繁杂，而且我们开发的过程中需要考虑更多的兼容性、扩展性。在JQuery中，已经将最常用的DOM操作方法进行了有效封装，并且不需要考虑浏览器的兼容性。

一、DOM简介

由于课程是基于JavaScript基础上完成的，这里我们不去详细地了解

DOM到底是什么，只需要知道下面几个基本概念：

（1）D表示的是页面文档Document、O表示对象，即一组含有独立特性的数据集合、M表示模型，即页面上的元素节点和文本节点。

（2）DOM有三种形式，即标准DOM、HTML DOM、CSS DOM，大部分都进行了一系列的封装，在JQuery中并不需要深刻理解它。

（3）树形结构用来表示DOM，就非常的贴切，大部分操作都是元素节点操作，还有少部分是文本节点操作。

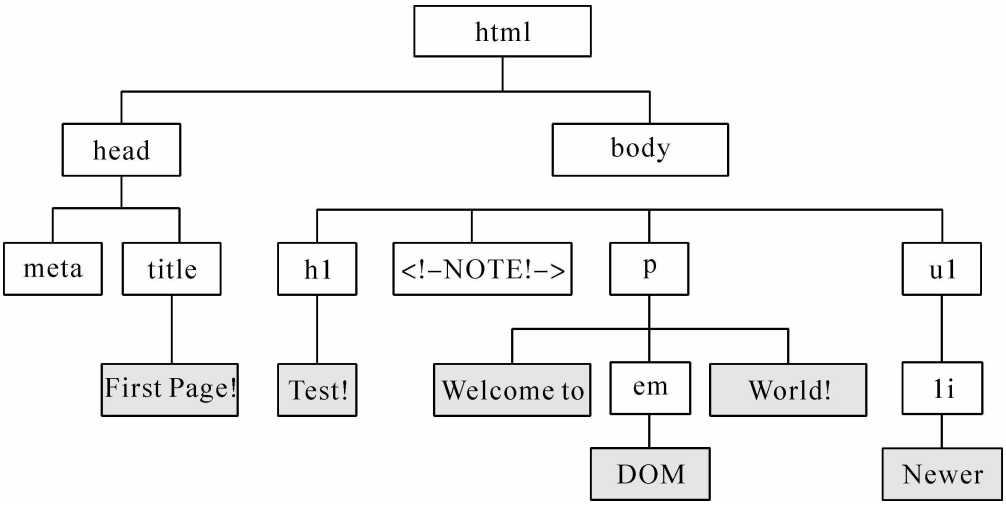

图5-1

二、设置元素及内容

通过前面所学习的各种选择器、过滤器来得到我们想要操作的元素。这个时候，我们就可以对这些元素进行DOM的操作。那么，最常用的操作就是对元素内容的获取和修改。

表5-1 html（）和text（）方法

方法名	描述
html（）	获取元素中HTML的内容。
html（value）	设置元素中HTML的内容。

text（）	获取元素中文本内容。
text（value）	设置原生中文本的内容。
val（）	获取表单中的文本的内容。
val（value）	设置表单中的文本的内容。

在常规的DOM元素中，我们可以使用html（）和text（）方法来获取内部的数据。通过html（）方法可以获取或设置html的内容，通过text（）可以获取或设置文本的内容。

$（'#box'）.html（）；//获取html的内容

$（'#box'）.text（）；//获取文本的内容，会自动清理html标签

$（'#box'）.html（'*www.li.cc*'）；//设置html的内容

$（'#box'）.text（'*www.li.cc*'）；//设置文本的内容，会自动转义html标签

注意：当我们使用html（）或text（）设置元素里的内容时，会清空原来的数据。而我们期望能够追加数据的话，需要先获取原本的数据。

$（'#box'）.html（$（'#box'）.html（）+ '*www.li.cc*'）；//追加数据

如果元素是表单的话，JQuery提供了val（）方法进行获取或设置内部的文本数据。

$（'input'）.val（）；//获取表单的内容

$（'input'）.val（'www.li.cc'）；//设置表单的内容

如果想设置多个选项的选定状态，比如下拉列表、单选复选框等，可以通过数组传递操作。

$（"input"）.val（［"check1"，"check2"，"radio1"］）；//value值是这些的将被选定

三、元素属性操作

除了对元素内容进行设置和获取，通过JQuery也可以对元素本身的属性进行操作，包括获取属性的属性值、设置属性的属性值，并且可以删除掉属性。

表5-2 attr（）和removeAttr（）

方法名	描述
attr（key）	获取某个元素key属性的属性值。
attr（key，value）	设置某个元素key属性的属性值。
attr（{key1：value2，key2：value2．..}）	设置某个元素多个key属性的属性值。
attr（key，function（index，value）{}）	设置某个元素key通过fn来设置。

```
$('div').attr('title'); //获取属性的属性值
$('div').attr('title', '我是域名'); //设置属性及属性值
$('div').attr('title', function () { //通过匿名函数返回属性值
    return '我是域名';
});
$('div').attr('title', function (index, value) {//可以接受
    return value + (index+1) + '，我是域名';
});
```

注意：attr（）方法里的function（）{}，可以不传参数。可以只传一个参数index，表示当前元素的索引（从0开始）。也可以传递两个参数index、value，第二个参数表示属性原本的值。

注意：JQuery中很多方法都可以使用function（）{}来返回出字符

串，比如html（）、text（）、val（）和上一章刚学过的is（）、filter（）方法。而如果又涉及多个元素集合，还可以传递index参数来获取索引值，并且可以使用第二个参数value（并不是所有方法都适合，有兴趣可以自己逐个尝试）。

```
$('div').html(function (index) {//通过匿名函数赋值，并传递inde
    return '我是' + (index+1) + '号div';
});
$('div').html(function (index, value) { //还可以实现追加内容
    return '我是' + (index+1) + '号div:'+value;
});
```

注意：我们也可以使用attr（）来创建id属性，但我们不建议这么做。这样会导致整个页面结构的混乱。当然也可以创建class属性，但后面会有一个语义更好的方法来代替attr（）方法，所以也不建议使用。

删除指定的属性，这个方法就不可以使用匿名函数，传递index和value均无效。

$('div').removeAttr('title'); //删除指定的属性

四、元素样式操作

元素样式操作包括直接设置CSS样式、增加CSS类别、类别切换、删除类别这几种操作方法。而从整个JQuery使用频率上来看，CSS样式的操作也是极高的，所以需要重点掌握。

表5-3 CSS操作方法

方法名	描述
css（name）	获取某个元素行内的CSS样式。
css（［name1，name2，name3］）	获取某个元素行内多个CSS样式。
css（name，value）	设置某个元素行内的CSS样式。
css（name，function（index，	

value））	设置某个元素行内的CSS样式。
css（{name1：value1，name2：value2}）	设置某个元素行内多个CSS样式。
addClass（class）	给某个元素添加一个CSS类。
addClass（class1 class2 class3．..）	给某个元素添加多个CSS类。
removeClass（class）	删除某个元素的一个CSS类。
removeClass（class1 class2 class3．..）	删除某个元素的多个CSS类。
toggleClass（class）	来回切换默认样式和指定样式。
toggleClass（class1 class2 class3．..）	同上。
toggleClass（class，switch）	来回切换样式的时候设置切换频率。
toggleClass（function（）{}）	通过匿名函数设置切换的规则。
toggleClass（function（）{}，switch）	在匿名函数设置时也可以设置频率。
toggleClass（function（i，c，s）{}，switch）	在匿名函数设置时传递三个参数。

$（'div'）.css（'color'）；//获取元素行内CSS样式的颜色

$（'div'）.css（'color'，'red'）；//设置元素行内CSS样式的颜色为红色

在CSS获取上，我们也可以获取多个CSS样式，而获取到的是一个对象数组，如果用传统方式进行解析需要使用for in遍历。

```
var box = $('div').css(['color', 'height', 'width']); /
for (var i in box) { //逐个遍历出来
    alert(i + ': ' + box[i]);
}
```

JQuery提供了一个遍历工具来专门处理这种对象数组，$.each（）方法，这个方法可以轻松地遍历对象数组。

```
$.each(box, function (attr, value) {//遍历JavaScript 原生态
    alert(attr + ': ' + value);
});
```

使用$.each（）可以遍历原生的JavaScript对象数组，如果是JQuery对象的数组怎么使用.each（）方法呢？

```
$('div').each(function (index, element) {//index 为索引，e
    alert(index + ': ' + element);
});
```

在需要设置多个样式的时候，我们可以传递多个CSS样式的键值对即可。

$（'div'）.css（{'background-color'：'#ccc'，'color'：'red'，'font-size'：'20px'}）；

如果想设置某个元素的CSS样式的值，但这个值需要计算。我们可以传递一个匿名函数。

```
$('div').css('width', function (index, value) {
    return (parseInt(value) - 500) + 'px';
});
```

除了行内CSS设置，我们也可以直接给元素添加CSS类，可以添加单个或多个，并且也可以删除它。

$（'div'）.addClass（'red'）；//添加一个CSS类

$（'div'）.addClass（'red bg'）；//添加多个CSS类

$（'div'）.removeClass（'bg'）；//删除一个CSS类

$（'div'）.removeClass（'red bg'）；//删除多个CSS类

我们还可以结合事件来实现CSS类的样式切换功能。

```
$（'div'）.click（function （） {//当点击后触发
    $（this）.toggleClass（'red size'）；//单个样式或多个样式均可
}）；
```

.toggleClass（）方法的第二个参数可以传入一个布尔值，true表示执行切换到class类，false表示执行默认class类（默认的是空class）。运用这个特性，我们可以设置切换的频率。

```
var count = 0；
    $（'div'）.click（function （） {//每点击两次切换一次red
    $（this）.toggleClass（'red'， count++ % 3 == 0）；
}）；
```

默认的CSS类切换只能是无样式和指定样式之间的切换，如果想实现样式1和样式2之间的切换还必须自己写一些逻辑。

```
$（'div'）.click（function （） {
    $（this）.toggleClass（'red size'）；//一开始切换到样式2
    if （$（this）.hasClass（'red'）） {//判断样式2 存在后
        $（this）.removeClass（'blue'）；//删除样式1
    } else {
        $（this）.toggleClass（'blue'）；//添加样式1，这里也可以add(
    }
    }）；
```

上面的方法较为繁琐，.toggleClass（）方法提供了传递匿名函数的方式，来设置你所需要切换的规则。

```
$（'div'）.click（function （） {
        $（this）.toggleClass（function （） { //传递匿名函数，返回
            return $（this）.hasClass（'red'） ? 'blue'： 'red si
        }）；
```

注意：上面虽然一句话实现了这个功能，但还是有一些小缺陷，因为原来的class类没有被删除，只不过被替代了而已。所以，需要改写一下。

```
$('div').click(function () {
    $(this).toggleClass(function () {
    if ($(this).hasClass('red')) {
        $(this).removeClass('red');
        return 'green';
    } else {
        $(this).removeClass('green');
        return 'red';
    }
    });
```

我们也可以在传递匿名函数的模式下增加第二个频率参数。

```
var count = 0;
$('div').click(function () {
    $(this).toggleClass(function () {
    return $(this).hasClass('red') ? 'blue': 'red size';
    }, count++ % 3 == 0); //增加了频率
});
```

对于.toggleClass（）传入匿名函数的方法，还可以传递index索引、class类两个参数以及频率布尔值，可以得到当前的索引、class类名和频率布尔值。

```
var count = 0;
$('div').click(function () {
    $(this).toggleClass(function (index, className, switch
    alert(index + ': ' + className + ': ' + switchBool); //得
    return $(this).hasClass('red') ? 'blue': 'red size';
    }, count++ % 3 == 0);
});
```

五、**CSS**方法

JQuery不但提供了CSS的核心操作方法，比如.css（）、.addClass（）等，还封装了一些特殊功能的CSS操作方法。下面我们分别来了解一下。

表5-4 width（）方法

方法名	描述
width（）	获取某个元素的长度。

width（value）	设置某个元素的长度。
width（function（index，width）{}）	通过匿名函数设置某个元素的长度。

```
$('div').width();  //获取元素的长度，返回的类型为number
$('div').width(500); //设置元素长度，直接传数值，默认加px
$('div').width('500pt');  //同上，设置了pt 单位
$('div').width(function (index, value) {//index 是索引，va
    return value - 500; //无须调整类型，直接计算
});
```

表5-5 height（）方法

方法名	描述
height（）	获取某个元素的长度。
height（value）	设置某个元素的长度。
height（function（index，width）{}）	通过匿名函数设置某个元素的长度。

```
$('div').height(); //获取元素的高度，返回的类型为number
$('div').height(500); //设置元素高度，直接传数值，默认加px
$('div').height('500pt'); //同上，设置了pt 单位
$('div').height(function (index, value) {//index 是索引，va
    return value - 1; //无需调整类型，直接计算
});
```

表5-6 内外边距和边框尺寸方法

方法名	描述
innerWidth（）	获取元素宽度，包含内边距padding。
innerHeight（）	获取元素高度，包含内边距

	padding。
outerWidth（）	获取元素宽度，包含边框border和内边距padding。
outerHeight（）	获取元素高度，包含边框border和内边距padding。
outerWidth（ture）	同上，且包含外边距。
outerHeight（true）	同上，且包含外边距。

alert（$（'div'）.width（））；//不包含

alert（$（'div'）.innerWidth（））；//包含内边距padding

alert（$（'div'）.outerWidth（））；//包含内边距padding+边框border

alert（$（'div'）.outerWidth（true））；//包含内边距padding+边框border+外边距margin

表5-7　元素偏移方法

方法名	描述
offset（）	获取某个元素相对于视口的偏移位置。
position（）	获取某个元素相对于父元素的偏移位置。
scrollTop（）	获取垂直滚动条的值。
scrollTop（value）	设置垂直滚动条的值。
scrollLeft（）	获取水平滚动条的值。

scrollLeft（value）	设置水平滚动条的值。

$（'strong'）.offset（）.left；//相对于视口的偏移

$（'strong'）.position（）.left；//相对于父元素的偏移

$（window）.scrollTop（）；//获取当前滚动条的位置

$（window）.scrollTop（300）；//设置当前滚动条的位置

第六章　DOM节点操作

教学要点：

1．创建节点；

2．插入节点；

3．包裹节点；

4．节点操作。

教学重点：

1．创建节点；

2．插入节点；

3．包裹节点；

4．节点操作。

教学难点：

理解什么是DOM及怎样去操作DOM。

开篇：DOM中有一个非常重要的功能，就是节点模型，也就是DOM中的“M”。页面中的元素结构就是通过这种节点模型来互相对应着的。我们只需要通过这些节点关系，就可以创建、插入、替换、克隆、删除等一系列的元素操作。

一、创建节点

为了使页面更加智能化，有时我们想动态地在html结构页面添加一个标签元素。那么在插入之前首先要做的动作就是：创建节点。

var box = $（' 节点 '）；//创建一个节点

$（'body'）.append（box）；//将节点插入到<body>元素内部

二、插入节点

在创建节点的过程中，其实我们已经演示怎么通过.append（）方法来插入一个节点。除了这个方法之外，JQuery还提供了其他几个方法来插入节点。

表6-1　内部插入节点方法

方法名	描述
append（content）	向指定元素内部后面插入节点content。
append（function（index，html）{}）	使用匿名函数向指定元素内部后面插入节点。
appendTo（content）	将指定元素移入到指定元素content内部后面。
prepend（content）	向指定元素content内部的前面插入节点。
prepend（function（index，html）{}）	使用匿名函数向指定元素内部的前面插入节点。
prependTo（content）	将指定元素移入到指定元素content内部前面。

```
    $（'div'）.append（'节点

'）；  //向div 内部插入strong 节点
    $（'div'）.append（function （index， html） {//使用匿名函数插入节
        return '节点

';
    }）；
    $（'span'）.appendTo（'div'）；//将span节点移入div 节点内
```

```
    $('span').appendTo($('div'));//同上
    $('div').prepend('节点
');  //将span 插入到div 内部的前面
    $('div').append(function (index, html) {//使用匿名函数，同上
        return '<span>节点</span>';
    });
    $('span').prependTo('div');//将span 移入div 内部的前面
    $('span').prependTo($('div'));//同上
```

表6-2 外部插入节点方法

方法名	描述
after（content）	向指定元素的外部后面插入节点content。
after（function（index，html）{}）	使用匿名函数向指定元素的外部后面插入节点。
before（content）	向指定元素的外部前面插入节点content。
before（function（index，html）{}）	使用匿名函数向指定元素的外部前面插入节点。
insertAfter（content）	将指定节点移到指定元素content外部的后面。
insertBefore（content）	将指定节点移到指定元素content外部的前面。

```
    $('div').after('<span>节点</span>');  //向div 的同级节点后面插入
    $('div').after(function (index, html) {//使用匿名函数，同上
        return '<span>节点</span>';
    });
    $('div').before('<span>节点</span>');  //向div 的同级节点前面插
    $('div').before(function (index, html) {//使用匿名函数，同上
        return '<span>节点</span>';
    });
```

```
$('span').insertAfter('div');  //将span 元素移到div 元素外部的
$('span').insertBefore('div'); //将span 元素移到div 元素外部的
```

三、包裹节点

JQuery提供了一系列方法用于包裹节点，那么包裹节点是什么意思呢？其实就是使用字符串代码将指定元素的代码包含着的意思。

表6-3　包裹节点

方法名	描述
wrap（html）	向指定元素包裹一层html代码。
wrap（element）	向指定元素包裹一层DOM对象节点。
wrap（function（index）{}）	使用匿名函数向指定元素包裹一层自定义内容。
unwrap（）	移除一层指定元素包裹的内容。
wrapAll（html）	用html将所有元素包裹到一起。
wrapAll（element）	用DOM对象将所有元素包裹在一起。
wrapInner（html）	向指定元素的子内容包裹一层html。
wrapInner（element）	向指定元素的子内容包裹一层DOM对象节点。
wrapInner（function（index）{}）	用匿名函数向指定元素的子内容包裹一层。

```
$('div').wrap('<strong></strong>'); //在div 外层包裹一层stron
```

```
$('div').wrap('<strong>123</strong>'); //包裹的元素可以带内容
$('div').wrap('<strong><em></em></strong>'); //包裹多个元素
$('div').wrap($('strong').get(0)); //也可以包裹一个原生DOM
$('div').wrap(function (index) {//匿名函数
   return '<strong></strong>';
});
$('div').unwrap(); //移除一层包裹内容，多个需移除多次
$('div').wrapAll('<strong></strong>'); //所有div 外面只包裹一
$('div').wrapAll($('strong').get(0)); //同上
$('div').wrapInner('<strong></strong>'); //包裹子元素内容
$('div').wrapInner($('strong').get(0)); //DOM 节点
$('div').wrapInner(function () {//匿名函数
   return '<strong></strong>';
});
```

注意：.wrap（）和.wrapAll（）的区别在前者把每个元素当成一个独立体，分别包含一层外层；后者将所有元素、一个整体作为一个独立体，只包含一层外层。这两种都是在外层包含，而.wrapInner（）在内层包含。

四、节点操作

除了创建、插入和包裹节点外，JQuery还提供了一些常规的节点操作方法：复制、替换和删除节点。

//复制节点

$（'body'）.append（$（'div'）.clone（true））；//复制一个节点添加到HTML中

注意：clone（true）参数可以为空，表示只复制元素和内容，不复制事件行为。而加上true参数的话，这个元素附带的事件处理行为也复制出来。

//删除节点

$（'div'）.remove（）；//直接删除div元素

注意：.remove（）不带参数时，删除前面对象选择器指定的元素。而.remove（）本身也可以带选择符参数的，比如：$（'div'）.remove（'#box'）；只删除id=box的div。

//保留事件的删除节点

$（'div'）.detach（）；//保留事件行为的删除

注意：.remove（）和.detach（）都是删除节点，而删除后本身方法可以返回当前被删除的节点对象，但区别在于前者在恢复时不保留事件行为，后者则会保留事件行为。

//清空节点

$（'div'）.empty（）；//删除掉节点里的内容

//替换节点

$（'div'）.replaceWith（'节点 '）；//将div替换成span元素

$（'节点 '）.replaceAll（'div'）；//同上

注意：节点被替换后，所包含的事件行为就全部消失了。

第七章　表单选择器

教学要点：

1．常规选择器；

2．表单选择器；

3．表单过滤器。

教学重点：

1．常规选择器；

2．表单选择器；

3．表单过滤器。

教学难点：

熟悉JQuery中表单选择器用法，多种表单选择器组合使用。

开篇：表单作为HTML中一种特殊的元素，操作方法较为多样性和特殊性，开发者不但可以使用之前的常规选择器或过滤器，也可以使用JQuery为表单专门提供的选择器和过滤器来准确地定位表单元素。

一、常规选择器

我们可以使用id、类（class）和元素名来获取表单字段，如果是表单元素，都必须含有name属性，还可以结合属性选择器来精确定位。

$（'input'）.val（）；//元素名定位，默认获取第一个

$（'input'）.eq（1）.val（）；//同上，默认获取第二个

$（'input［type=password］'）.val（）；//选择type为password的字段

$（'input［name=user］'）.val（）；//选择name为user的字段

那么对于id和类（class）用法比较类似，也可以结合属性选择器来精确定位，这里我们不再重复。对于表单中的其他元素名比如textarea、select和button等，原理一样，这里不再重复。

二、表单选择器

虽然可以使用常规选择器来对表单的元素进行定位，但有时还是不能满足开发者灵活多变的需求。所以，JQuery为表单提供了专用的选择器。

表7-1　表单选择器

方法名	描述	返回
：input	选取所有input、textarea、select和button元素	集合元素
：text	选择所有单行文本框，即type=text	集合元素
：password	选择所有密码框，即type=password	集合元素
：radio	选择所有单选框，即type=radio	集合元素
：checkbox	选择所有复选框，即type=checkbox	集合元素
：submit	选取所有提交按钮，即type=submit	集合元素
：reset	选取所有重置按钮，即type=reset	集合元素
：image	选取所有图像按钮，即type=image	集合元素

：button	选择所有普通按钮，即button元素	集合元素
：file	选择所有文件按钮，即type=file	集合元素
：hidden	选择所有不可见字段，即type=hidden	集合元素

$（'：input'）.size（）；//获取所有表单字段元素

$（'：text）.size（）；//获取单行文本框元素

$（'：password'）.size（）；//获取密码栏元素

$（'：radio）.size（）；//获取单选框元素

$（'：checkbox）.size（）；//获取复选框元素

$（'：submit）.size（）；//获取提交按钮元素

$（'：reset）.size（）；//获取重置按钮元素

$（'：image）.size（）；//获取图片按钮元素

$（'：file）.size（）；//获取文件按钮元素

$（'：button）.size（）；//获取普通按钮元素

$（'：hidden）.size（）；//获取隐藏字段元素

注意：这些选择器都是返回元素集合，如果想获取某一个指定的元素，最好结合一下属性选择器。比如：

$（'：text［name=user］）.size（）；//获取单行文本框name=user的元素

三、表单过滤器

JQuery提供了四种表单过滤器，分别在是否可以用、是否选定来进行表单字段的筛选过滤。

表7-2　表单过滤器

方法名	描述	返回
：enabled	选取所有可用元素	集合元素
：disabled	选取所有不可用元素	集合元素
：checked	选取所有被选中的元素，单选和复选字段	集合元素
：selected	选取所有被选中的元素，下拉列表	集合元素

$（'：enabled'）.size（）；//获取可用元素

$（'：disabled）.size（）；//获取不可用元素

$（'：checked）.size（）；//获取单选、复选框中被选中的元素

$（'：selected）.size（）；//获取下拉列表中被选中的元素

第八章　基础事件

教学要点：

1．绑定事件；

2．简写事件；

3．复合事件。

教学重点：

1．绑定事件；

2．简写事件；

3．复合事件。

教学难点：

理解什么是事件、在JQuery中的常用事件。

开篇：JavaScript有一个非常重要的功能，就是事件驱动。当页面完全加载后，用户通过鼠标或键盘触发页面中绑定事件的元素即可触发。JQuery为开发者更有效率的编写事件行为，封装了大量有益的事件方法供我们使用。

一、绑定事件

在JavaScript课程的学习中，我们掌握了很多使用的事件，常用的事件有：click、dblclick、mousedown、mouseup、mousemove、mouseover、mouseout、change、select、submit、keydown、keypress、keyup、blur、focus、load、resize、scroll、error。那么，还有更多的事件可以参考手册中的事件部分。

JQuery通过.bind（）方法来为元素绑定这些事件。可以传递三个参数：bind（type，［data］，fn），type表示一个或多个类型的事件名字

符串；［data］是可选的，作为event.data属性值传递一个额外的数据，这个数据是一个字符串、一个数字、一个数组或一个对象；fn表示绑定到指定元素的处理函数。

```
//使用点击事件
$（'input'）.bind（'click'， function （） {//点击按钮后执行匿名函
    alert（'点击！'）；
}）；
//普通处理函数
$（'input'）.bind（'click'， fn）；//执行普通函数式无须圆括号
function fn（） {
    alert（'点击！'）；
}
//可以同时绑定多个事件
$（'input'）.bind（'mouseout mouseover'， function （） { //移入
    $（'div'）.html（function （index， value） {
        return value + '1'；
    }）；
}）；
//通过对象键值对绑定多个参数
$（'input'）.bind（{//传递一个对象
    'mouseout'： function （） {//事件名的引号可以省略
        alert（'移出'）；
    }，
    'mouseover'： function （） {
        alert（'移入'）；
    }
}）；
//使用unbind 删除绑定的事件
$（'input'）.unbind（）；//删除所有当前元素的事件
//使用unbind 参数删除指定类型事件
$（'input'）.unbind（'click'）；//删除当前元素的click 事件
//使用unbind 参数删除指定处理函数的事件
function fn1（） {
    alert（'点击1'）；
}
function fn2（） {
    alert（'点击2'）；
}
$（'input'）.bind（'click'， fn1）；
$（'input'）.bind（'click'， fn2）；
$（'input'）.unbind（'click'， fn1）；//只删除fn1 处理函数的事件
```

二、简写事件

为了使开发者更加方便的绑定事件，JQuery封装了常用的事件以便

节约更多的代码。我们称它为简写事件。

表8-1　简写事件绑定方法

方法名	触发条件	描述
click（fn）	鼠标	触发每一个匹配元素的click（单击）事件
dblclick（fn）	鼠标	触发每一个匹配元素的dblclick（双击）事件
mousedown（fn）	鼠标	触发每一个匹配元素的mousedown（点击后）事件
mouseup（fn）	鼠标	触发每一个匹配元素的mouseup（点击弹起）事件
mouseover（fn）	鼠标	触发每一个匹配元素的mouseover（鼠标移入）事件
mouseout（fn）	鼠标	触发每一个匹配元素的mouseout（鼠标移出）事件
mousemove（fn）	鼠标	触发每一个匹配元素的mousemove（鼠标移动）事件
mouseenter（fn）	鼠标	触发每一个匹配元素的mouseenter（鼠标穿过）事件
mouseleave（fn）	鼠标	触发每一个匹配元素的mouseleave（鼠标穿出）事件
keydown（fn）	键盘	触发每一个匹配元素的keydown（键盘按下）事件
		触发每一个匹配元素的

keyup（fn）	键盘	keyup（键盘按下弹起）事件
keypress（fn）	键盘	触发每一个匹配元素的keypress（键盘按下）事件
unload（fn）	文档	当卸载本页面时绑定一个要执行的函数
resize（fn）	文档	触发每一个匹配元素的resize（文档改变大小）事件
scroll（fn）	文档	触发每一个匹配元素的scroll（滚动条拖动）事件
focus（fn）	表单	触发每一个匹配元素的focus（焦点激活）事件
blur（fn）	表单	触发每一个匹配元素的blur（焦点丢失）事件
focusin（fn）	表单	触发每一个匹配元素的focusin（焦点激活）事件
focusout（fn）	表单	触发每一个匹配元素的focusout（焦点丢失）事件
select（fn）	表单	触发每一个匹配元素的select（文本选定）事件
change（fn）	表单	触发每一个匹配元素的change（值改变）事件
submit（fn）	表单	触发每一个匹配元素的submit（表单提交）事件

注意：这里封装的大部分方法都比较好理解，我们没必要一一演示确认，重点看几个需要注意区分的简写方法。

.mouseover（）和.mouseout（）表示鼠标移入和移出的时候触发。那么JQuery还封装了另外一组：.mouseenter（）和.mouseleave（）表示鼠标穿过与穿出的时候触发。那么这两组本质上有什么区别呢？手册上的说明是：.mouseenter（）和.mouseleave（）这组穿过子元素不会触发，而.mouseover（）和.mouseout（）则会触发。

```
//HTML页面设置
<div style="width：200px；height：200px；background：green；">
    <p style="width：100px；height：100px；background：red；">
</div>
<strong></strong>
//mouseover移入
$（'div'）.mouseover（function （） {//移入div 会触发，移入p 再触发
    $（'strong'）.html（function （index， value） {
        return value+'1'；
    }）；
}）；
//mouseenter穿过
$（'div'）.mouseenter（function （） {//穿过div 或者p
    $（'strong'）.html（function （index， value） {//在这个区域只
        return value+'1'；
    }）；
}）；
//mouseout移出
$（'div'）.mouseout（function （） { //移出p 会触发，移出div 再触发
    $（'strong'）.html（function （index， value） {
        return value+'1'；
    }）；
}）；
//mouseleave穿出
$（'div'）.mouseleave（function （） {//移出整个div 区域触发一次
    $（'strong'）.html（function （index， value） {
        return value+'1'；
    }）；
}）；
.keydown（）、.keyup（）返回的是键码，而.keypress（）返回的是字符编码
$（'input'）.keydown（function （e） {
    alert（e.keyCode）；//按下a 返回65
}）；
$（'input'）.keypress（function （e） {
    alert（e.charCode）；//按下a 返回97
}）；
```

注意：e.keyCode和e.charCode在两种事件互换也会产生不同的效果，除了字符还有一些非字符键的区别。

.focus（）和.blur（）分别表示光标激活和丢失，事件触发时机是当前元素。而.focusin（）和.focusout（）也表示光标激活和丢失，但事件触发时机可以是子元素。

```
//HTML部分
<div style="width：200px；height：200px；background：red；">
    <input type="text" value="" />
</div>
<strong></strong>
//focus光标激活
$（'input'）.focus（function （） {//当前元素触发
    $（'strong'）.html（'123'）；
}）；
//focusin光标激活
$（'div'）.focusin（function （） {//绑定的是div 元素，子类input 触
    $（'strong'）.html（'123'）；
}）；
```

注意：.blur（）和.focusout（）表示光标丢失，和激活类似，一个必须当前元素触发，另一个可以是子元素触发。

三、复合事件

JQuery提供了许多最常用的事件效果，组合一些功能实现了一些复合事件，比如切换功能、智能加载等。

表8-2　复合事件

方法名	描述
ready（fn）	当DOM加载完毕触发事件
hover（［fn1，］fn2）	当鼠标移入触发第一个fn1，移出触发fn2
toggle（fn1，fn2［，fn3．.］）	已废弃，当鼠标点击触发fn1，再点击触发fn2．..

```
//背景移入移出切换效果
$('div').hover(function () {
    $(this).css('background', 'black'); //mouseenter 效果
}, function () {
    $(this).css('background', 'red'); //mouseleave 效果，可省
});
```

注意：.hover（）方法是结合了.mouseenter（）方法和.mouseleva（）方法，并非.mouseover（）和.mouseout（）方法。

.toggle（）这个方法比较特殊，这个方法有两层含义：第一层含义就是已经被1.8版废用、1.9版删除的用法，也就是点击切换复合事件的用法；第二层含我将会在动画那章讲解到。既然废弃掉了，就不应该使用。被删除的原因是：以减少混乱和提高潜在的模块化程度。

但你又非常想用这个方法，并且不想自己编写类似的功能，可以下载jquery-migrate.js文件，来向下兼容已被删除掉的方法。

```
//背景点击切换效果（1.9 版删除了）
<script type="text/javascript" src="jquery-migrate-1.2.1.js">
$('div').toggle(function () {//第一次点击切换
    $(this).css('background', 'black');
}, function () {//第二次点击切换
    $(this).css('background', 'blue');
}, function () {//第三次点击切换
    $(this).css('background', 'red');
});
```

注意：由于官方已经删除掉这个方法，所以也是不推荐使用的，如果在不基于向下兼容的插件JS。我们可以自己实现这个功能。

```
var flag = 1; //计数器
$('div').click(function () {
    if (flag == 1) {//第一次点击
        $(this).css('background', 'black');
        flag = 2;
    } else if (flag == 2) {//第二次点击
        $(this).css('background', 'blue');
        flag = 3
    } else if (flag == 3) {//第三次点击
        $(this).css('background', 'red');
        flag = 1
```

```
    }
});
```

第九章　事件对象

教学要点：

1．事件对象；

2．冒泡和默认行为。

教学重点：

1．事件对象；

2．冒泡和默认行为。

教学难点：

事件冒泡。

开篇：JavaScript在事件处理函数中默认传递了event对象，也就是事件对象。但由于浏览器的兼容性，开发者总是会做兼容方面的处理。JQuery在封装的时候，解决了这些问题，并且还创建了一些非常好用的属性和方法。

一、事件对象

事件对象就是event对象，通过处理函数默认传递接受。之前处理函数的e就是event事件对象，event对象有很多可用的属性和方法，我们在JavaScript课程中已经详细地了解过这些常用的属性和方法。这里，我们再一次演示一下。

```
//通过处理函数传递事件对象
$（'input'）.bind（'click', function （e） { //接受事件对象参数
   alert（e）；
}）；
```

表9-1　event对象的属性

属性名	描述
type	获取这个事件的事件类型，如click
target	获取绑定事件的DOM元素
data	获取事件调用时的额外数据
relatedTarget	获取移入移出目标点离开或进入的那个DOM元素
currentTarget	获取冒泡前触发的DOM元素，等同于this
pageX/pageY	获取相对于页面原点的水平/垂直坐标
screenX/screenY	获取显示器屏幕位置的水平/垂直坐标（非JQuery封装）
clientX/clientY	获取相对于页面视口的水平/垂直坐标（非JQuery封装）
result	获取上一个相同事件的返回值
timeStamp	获取事件触发的时间戳
which	获取鼠标的左中右键（1，2，3），或获取键盘按键
altKey /shiftKey /ctrlKey /metaKey	获取是否按下了alt、shift、ctrl（这三个非JQuery封装）或meta键（IE原生meta键，JQuery做了封装）

```
//通过event.type 属性获取触发事件名
$（'input'）.click（function （e） {
alert（e.type）；
}）；
```

```
//通过event.target 获取绑定的DOM 元素
$（'input'）.click（function （e） {
alert（e.target）；
}）；
//通过event.data 获取额外数据，可以是数字、字符串、数组、对象
$（'input'）.bind（'click'， 123， function （） {//传递data 数据
alert（e.data）；//获取数字数据
}）；
```

注意：如果字符串就传递：'123'，如果是数组就传递：［123，'abc'］，如果是对象就传递：

```
{user： 'Lee'， age： 100}。数组的调用方式是：e.data［1］，对象的调用
//event.data获取额外数据，对于封装的简写事件也可以使用
$（'input'）.click（{user： 'Lee'， age： 100}，function （e） {
alert（e.data.user）；
}）；
```

注意：键值对的键可以加上引号，也可以不加；在调用的时候也可以使用数组的方式：

```
alert（e.data［'user'］）；
//获取移入到div 之前的那个DOM 元素
$（'div'）.mouseover（function （e） {
alert（e.relatedTarget）；
}）；
//获取移出div 之后到达最近的那个DOM 元素
$（'div'）.mouseout（function （e） {
alert（e.relatedTarget）；
}）；
//获取绑定的那个DOM 元素，相当于this，区别于event.target
$（'div'）.click（function （e） {
alert（e.currentTarget）；
}）；
```

注意：event.target得到的是触发元素的DOM，event.currentTarget得到的是监听元素的DOM。而this也是得到监听元素的DOM。

```
//获取上一次事件的返回值
$（'div'）.click（function （e） {
return '123'；
}）；
$（'div'）.click（function （e） {
alert（e.result）；
}）；
```

```
//获取当前的时间戳
$（'div'）.click（function （e） {
alert（e.timeStamp）；
}）；
//获取鼠标的左中右键
$（'div'）.mousedown（function （e） {
alert（e.which）；
}）；
//获取键盘的按键
$（'input'）.keyup（function （e） {
alert（e.which）；
}）；
//获取是否按下了ctrl 键，meta 键不存在，导致无法使用
$（'input'）.click（function （e） {
alert（e.ctrlKey）；
}）；
//获取触发元素鼠标当前的位置
$（document）.click（function （e） {
alert（e.screenY+ ', ' + e.pageY + ', ' + e.clientY）；
}）；
```

二、冒泡和默认行为

如果在页面中重叠了多个元素，并且重叠的这些元素都绑定了同一个事件，那么就会出现冒泡问题。

```
//HTML页面
<div style="width：200px；height：200px；background：red；">
    <input type="button" value="按钮" />
</div>
//三个不同元素触发事件
$（'input'）.click（function （） {
    alert（'按钮被触发了！'）；
}）；
$（'div'）.click（function （） {
    alert（'div层被触发了！'）；
}）；
$（document）.click（function （） {
    alert（'文档页面被触发了！'）；
}）；
```

注意：当我们点击文档的时候，只触发文档事件；当我们点击div层时，触发了div和文档两个；当我们点击按钮时，触发了按钮、div和文档。触发的顺序是从小范围到大范围。这就是所谓的冒泡现象，一层一层往上。

JQuery提供了一个事件对象的方法：event.stopPropagation（）。这个方法设置到需要触发的事件上时，所有上层的冒泡行为都将被取消。

```
$（'input'）.click（function （e） {
    alert（'按钮被触发了！'）；
    e.stopPropagation（）；
}）；
```

网页中的元素，在操作的时候会有自己的默认行为。比如：右击文本框输入区域，会弹出系统菜单、点击超链接会跳转到指定页面、点击提交按钮会提交数据。

```
$（'a'）.click（function （e） {
    e.preventDefault（）；
}）
//禁止提交表单跳转
$（'form'）.submit（function （e） {
    e.preventDefault（）；
}）；
```

注意：如果想让上面的超链接同时阻止默认行为且禁止冒泡行为，可以把两个方法同时写上：event.stopPropagation（）和event.preventDefault（）。这两个方法如果需要同时启用的时候，还有一种简写方案代替，就是直接return false。

```
$（'a'）.click（function （e） {
    return false；
}）；
```

表9-2　冒泡和默认行为的一些方法

方法名	描述
prevent Default（）	取消某个元素的默认行为
isDefault Prevented（）	判断是否调用了preventDefault（）方法
stop Propagation（）	取消事件冒泡

isPropagation Stopped（）	判断是否调用了 stopPropagation（）方法
stop Immediate Propagation（）	取消事件冒泡，并取消该事件的后续事件处理函数
isImmediate Propagation Stopped（）	判断是否调用了 stopImmediatePropagation（）方法

```
//判断是否取消了元素的默认行为
$（'input'）.keyup（function （e） {
    e.preventDefault（）；
    alert（e.isDefaultPrevented（））；
}）；
//取消冒泡并取消后续事件处理函数
$（'input'）.click（function （e） {
    alert（'input'）；
    e.stopImmediatePropagation（）；
}）；
$（'input'）.click（function （） {
    alert（'input2'）；
}）；
$（document）.click（function （） {
    alert（'document'）；
}）；
//判断是否调用了stopPropagation（）方法
$（'input'）.click（function （e） {
    e.stopPropagation（）；
    alert（e.isPropagationStopped（））；
}）；
//判断是否执行了stopImmediatePropagation（）方法
$（'input'）.click（function （e） {
    e.stopImmediatePropagation（）；
    alert（e.isImmediatePropagationStopped（））；
}）；
```

第十章　高级事件

教学要点：

1．模拟操作；

2．命名空间；

3．事件委托；

4．On、Off和One。

教学重点：

1．模拟操作；

2．命名空间；

3．事件委托；

4．On、Off和One。

教学难点：

命名空间和事件委托。

开篇：JQuery不但封装了大量常用的事件处理，还提供了不少高级事件方便开发者使用。比如模拟用户触发事件、事件委托事件和统一整合的on与off，以及仅执行一次的one方法。这些方法大大降低了开发者难度，提升了开发者的开发体验。

一、模拟操作

在事件触发的时候，有时我们需要一些模拟用户行为的操作。例如，当网页加载完毕后自行点击一个按钮触发一个事件，而不是由用户去点击。

```
//点击按钮事件
$('input').click(function () {
    alert('我的第一次点击来自模拟!');
});
//模拟用户点击行为
$('input').trigger('click');
//可以合并两个方法
$('input').click(function () {
    alert('我的第一次点击来自模拟!');
}).trigger('click');
```

有时在模拟用户行为的时候，我们需要给事件执行传递参数，这个参数类似于event.data的额外数据，可以是数字、字符串、数组、对象。

```
$('input').click(function (e, data1, data2) {
    alert(data1 + ', ' + data2);
}).trigger('click', ['abc', '123']);
```

注意：当传递一个值的时候，直接传递即可。当两个值以上时，需要在前后用中括号包含起来。但不能认为是数组形式，下面给出一个复杂的说明。

```
$('input').click(function (e, data1, data2) {
    alert(data1.a + ', ' + data2[1]);
}).trigger('click', [{'a': '1', 'b': '2'}, ['123','456
```

除了通过JavaScript事件名触发，也可以通过自定义的事件触发。所谓自定义事件，其实就是一个被.bind()绑定的任意函数。

```
$('input').bind('myEvent', function () {
    alert('自定义事件!');
}).trigger('myEvent');
```

.trigger()方法提供了简写方案，只要想让某个事件执行模拟用户行为，直接再调用一个空的同名事件即可。

```
$('input').click(function () {
    alert('我的第一次点击来自模拟!');
}).click(); //空的click()执行的是trigger()
```

这种便捷的方法，JQuery几乎所有常用的事件都提供了。

表10-1

blur	focusin	mousedown	resize
change	focusout	mousenter	scroll
click	keydown	mouseleave	select
dblclick	keypress	mousemove	submit
error	keyup	mouseout	unload
focus	load	mouseover	

JQuery还提供了另外一个模拟用户行为的方法：.triggerHandler（）；这个方法的使用和.trigger（）方法一样。

```
$（'input'）.click（function （） {
    alert（'我的第一次点击来自模拟！'）；
}）.triggerHandler（'click'）；
```

在常规的使用情况下，两者几乎没有区别，都是模拟用户行为，也可以传递额外参数。但在某些特殊情况下，就产生了差异：

（1）triggerHandler（）方法并不会触发事件的默认行为，而.trigger（）会。

```
$（'form'）.trigger（'submit'）；//模拟用户执行提交，并跳转到执行页面
$（'form'）.triggerHandler（'submit'）；//模拟用户执行提交，并阻止的
如果我们希望使用.trigger（）来模拟用户提交，并且阻止事件的默认行为，则需
$（'form'）.submit（function （e） {
    e.preventDefault（）；//阻止默认行为
}）.trigger（'submit'）；
```

（2）triggerHandler（）方法只会影响第一个匹配到的元素，而.trigger（）会影响所有。

（3）triggerHandler（）方法会返回当前事件执行的返回值，如果没有返回值，则返回undefined；而.trigger（）则返回当前包含事件触发元素的JQuery对象（方便链式连缀调用）。

```
alert（$（'input'）.click（function （） {
    return 123;
}）.triggerHandler（'click'））；//返回123，没有return 返回
```

（4）trigger（）在创建事件的时候，会冒泡。但这种冒泡是自定义事件才能体现出来，是JQuery扩展于DOM的机制，并非DOM特性。而.triggerHandler（）不会冒泡。

```
var index = 1;
$（'div'）.bind（'myEvent', function（）{
    alert（'自定义事件' + index）；
    index++;
    }）；
$（'.div3'）.trigger（"myEvent"）；
```

二、命名空间

有时，我们想对事件进行移除。但对于同名同元素绑定的事件移除往往比较麻烦，这个时候，可以使用事件的命名空间解决。

```
$（'input'）.bind（'click.abc', function （） {
    alert（'abc'）；
}）；
$（'input'）.bind（'click.xyz', function （） {
    alert（'xyz'）；
}）；
$（'input'）.unbind（'click.abc'）；//移除click事件中命名空间为abc
```

注意：也可以直接使用（'.abc'），这样，可以移除相同命名空间的不同事件。对于模拟操作.trigger（）和.triggerHandler（），其用法也是一样的。

$（'input'）.trigger（'click.abc'）；

三、事件委托

什么是事件委托？用现实中的理解就是：有100个学生同时在某天中午收到快递，但这100个学生不可能同时站在学校门口等，那么都会委托门卫去收取，然后再逐个交给学生。而在JQuery中，我们通过事件冒泡的特性，让子元素绑定的事件冒泡到父元素（或祖先元素）上，然后再进行相关处理即可。

如果一个企业级应用做报表处理，表格有2000行，每一行都有一个按钮处理。如果用之前的.bind（）处理，那么就需要绑定2000个事件，就好比2000个学生同时站在学校门口等快递，不断会堵塞路口，还会发生各种意外。这种情况放到页面上也是一样，可能导致页面极度变慢或直接异常。而且，2000个按钮使用ajax分页的话，.bind（）方法无法动态绑定尚未存在的元素。这就好比新转学的学生，快递员无法验证他的身份，就可能收不到快递。

```
//HTML部分
<div style="background：red；width：200px；height：200px；" id=
    <input type="button" value="按钮" class="button" />
</div>
//使用.bind（）不具备动态绑定功能，只有点击原始按钮才能生成
$（'.button'）.bind（'click'， function （） {
    $（this）.clone（）.appendTo（'#box'）；
}）；
//使用.live（）具备动态绑定功能，JQuery1.3 使用，JQuery1.7 之后废弃，
$（'.button'）.live（'click'， function （） {
    $（this）.clone（）.appendTo（'#box'）；
}）；
```

.live（）原理就是把click事件绑定到祖先元素$（document）上，而只需要给$（document）绑定一次即可，而非2000次。然后，就可以处理后续动态加载的按钮的单击事件。在接受任何事件时，$（document）对象都会检查事件类型（event.type）和事件目标（event.target）。如果click事件是.button，那么就执行委托给它的处理程序。.live（）方法已经被删除，无法使用了。需要测试使用的话，需要引入向下兼容插件。

```
//.live（）无法使用链接连缀调用，因为参数的特性导致
$（'#box'）.children（0）.live（'click'， function （） {
    $（this）.clone（）.appendTo（'#box'）；
}）；
```

在上面的例子中，我们使用了.clone（）克隆。其实如果想把事件行为复制过来，我们只需要传递true即可：.clone（true）。这样也能实现类似事件委托的功能，但原理截然不同。一个是复制事件行为，另一个是事件委托。而在非克隆操作下，此类功能只能使用事件委托。

```
$（'.button'）.live（'click'， function （） {
    $（'<input type="button" value="复制的" class="button" />'）
}）；
```

当我们需要停止事件委托的时候，可以使用.die（）来取消。

```
$（'.button'）.die（'click'）；
```

由于.live（）和.die（）在JQuery1.4.3版本中废弃了，之后推出语义清晰、减少冒泡传播层次又支持链接连缀调用方式的方法：.delegate（）和.undelegate（）。但这个方法在JQuery1.7版本中被.on（）方法整合替代了。

```
$（'#box'）.delegate（'.button'， 'click'， function （） {
    $（this）.clone（）.appendTo（'#box'）；
}）；
$（'#box'）.undelegate（'.button'，'click'）；
//支持连缀调用方式
$（'div'）.first（）.delegate（'.button'， 'click'， function （
    $（this）.clone（）.appendTo（'div：first'）；
}）；
```

注意：.delegate（）需要指定父元素，第一个参数是当前元素，第二个参数是事件方式，第三个参数是执行函数。和.bind（）方法一样，可以传递额外参数。.undelegate（）和.unbind（）方法一样可以直接删除所有事件，比如：.undelegate（'click'）。也可以删除命名空间的事件，比如：.undelegate（'click.abc'）。

注意：.live（）和.delegate（）与.bind（）方法一样都是事件绑定，那么区别也很明显，用途上遵循以下两个规则：①在DOM中很多元素绑定相同事件时；②在DOM中尚不存在即将生成的元素绑定事件时。我们推荐使用事件委托的绑定方式，否则推荐使用.bind（）的普通绑定。

四、**on**、**off**和**one**

目前绑定事件和解绑的方法有三组共六个。由于这三组的共存可能会造成一定的混乱，为此JQuery1.7以后推出了.on（）和.off（）方法彻底摒弃前面三组。

```
//替代.bind（）方式
$（'.button'）.on（'click'， function （） {
    alert（'替代.bind（）'）；
}）；
//替代.bind（）方式，并使用额外数据和事件对象
```

```
$（'.button'）.on（'click'， {user： 'Lee'}， function （e） {
    alert（'替代.bind（）' + e.data.user）；
}）；
//替代.bind（）方式，并绑定多个事件
$（'.button'）.on（'mouseover mouseout'， function （） {
    alert（'替代.bind（）移入移出！'）；
}）；
//替代.bind（）方式，以对象模式绑定多个事件
$（'.button'）.on（{
    mouseover： function （） {
        alert（'替代.bind（）移入！'）；
    }，
    mouseout： function （） {
        alert（'替代.bind（）移出！'）；
    }
}）；
//替代.bind（）方式，阻止默认行为并取消冒泡
$（'form'）.on（'submit'， function （） {
    return false；
}）；
或
$（'form'）.on（'submit'， false）；
//替代.bind（）方式，阻止默认行为
$（'form'）.on（'submit'， function （e） {
    e.preventDefault（）；
}）；
//替代.bind（）方式，取消冒泡
$（'form'）.on（'submit'， function （e） {
    e.stopPropagation（）；
}）；
//替代.unbind（）方式，移除事件
$（'.button'）.off（'click'）；
$（'.button'）.off（'click'， fn）；
$（'.button'）.off（'click.abc'）；
//替代.live（）和.delegate（），事件委托
$（'#box'）.on（'click'， '.button'， function （） {
    $（this）.clone（）.appendTo（'#box'）；
}）；
//替代.die（）和.undelegate（），取消事件委托
$（'#box'）.off（'click'， '.button'）；
```

注意：和之前方式一样，事件委托和取消事件委托也有各种搭配方式，比如额外数据、命名空间等，这里不再赘述。

不管是.bind（）还是.on（），绑定事件后都不是自动移除事件的，需要通过.unbind（）和.off（）来手工移除。JQuery提供了.one（）

方法，绑定元素执行完毕后自动移除事件，仅触发一次的事件。

```
//类似于.bind（）只触发一次
$（'.button'）.one（'click'， function （） {
    alert（'one仅触发一次！'）；
}）；
//类似于.delegate（）只触发一次
$（'#box）.one（'click'， 'click'， function （） {
    alert（'one仅触发一次！'）；
}）；
```

第十一章　动画效果

教学要点：

1．显示、隐藏；

2．滑动、卷动；

3．淡入、淡出；

4．自定义动画；

5．列队动画方法；

6．动画相关方法；

7．动画全局属性。

教学重点：

1．显示、隐藏；

2．滑动、卷动；

3．淡入、淡出；

4．自定义动画；

5．列队动画方法；

6．动画相关方法；

7．动画全局属性。

教学难点：

事件冒泡。

开篇：在以前很长一段时间里，网页上的各种特效还需要采用flash在进行。但最近几年里，我们已经很少看到这种情况了，绝大部分已经使用JavaScript动画效果来取代flash。这里说的取代是网页特效部分，而不是动画。网页特效比如渐变菜单、渐进显示、图片轮播等；而动画比如故事情节广告、MV等。

一、显示、隐藏

JQuery中的显示方法为：.show（），隐藏方法为：.hide（）。在无参数的时候，只是硬性的显示内容和隐藏内容。

```
$（'.show'）.click（function （） {//显示
    $（'#box'）.show（）；
}）；
$（'.hide'）.click（function （） {//隐藏
    $（'#box'）.hide（）；
}）；
```

注意：.hide（）方法其实就是在行内设置CSS代码：display：none。 而.show（）方法要根据原来元素是区块还是内联来决定，如果是区块，则设置CSS代码：display：block。如果是内联，则设置CSS代码：display：inline。

在.show（）和.hide（）方法中可以传递一个参数，这个参数以毫秒（1000毫秒等于1秒钟）来控制速度。并且包含了匀速变大变小，以及透明度变换。

```
$（'.show'）.click（function （） {
    $（'#box'）.show（1000）；//显示用了1 秒
}）；
$（'.hide'）.click（function （） {
    $（'#box'）.hide（1000）；//隐藏用了1 秒
}）；
```

除了直接使用毫秒来控制速度外，JQuery还提供了三种预设速度参数字符串：slow、normal和fast，分别对应600毫秒、400毫秒和200毫秒。

```
$（'.show'）.click（function （） {
    $（'#box'）.show（'fast'）；//200 毫秒
}）；
$（'.hide'）.click（function （） {
```

```
    $('#box').hide('slow'); //600 毫秒
});
```

注意：不管是传递毫秒数还是传递预设字符串，如果不小心传递错误或者传递空字符串。那么它将采用默认值：400毫秒。

```
$('.show').click(function () {
    $('#box').show(''); //默认400 毫秒
});
//使用.show()和.hide()的回调函数，可以实现列队动画效果
$('.show').click(function () {
    $('#box').show('slow', function () {
        alert('动画持续完毕后，执行我!');
    });
});
//列队动画，使用函数名调用自身
$('.show').click(function () {
    $('div').first().show('fast', function showSpan() {
        $(this).next().show('fast', showSpan);
    });
});
//列队动画，使用arguments.callee 匿名函数自调用
$('.hide').click(function () {
    $('div').last().hide('fast', function() {
        $(this).prev().hide('fast', arguments.callee);
    });
});
```

我们在使用.show（）和.hide（）的时候，如果需要一个按钮切换操作，需要进行一些条件判断。而JQuery提供给我们一个类似功能的独立方法：.toggle（）。

```
$('.toggle').click(function () {
    $(this).toggle('slow');
});
```

二、滑动、卷动

JQuery提供了一组改变元素高度的方法：.slideUp（）、.slideDown（）和.slideToggle（）。顾名思义，向上收缩（卷动）和向下展开（滑动）。

```
$('.down').click(function () {
    $('#box').slideDown();
```

```
});
$('.up').click(function () {
    $('#box').slideUp();
});
$('.toggle').click(function () {
    $('#box').slideToggle();
});
```

注意：滑动、卷动效果和显示、隐藏效果一样，具有相同的参数。

三、淡入、淡出

JQuery提供了一组专门用于透明度变化的方法：.fadeIn（）和.fadeOut（），分别表示淡入、淡出。当然，还有一个自动切换的方法：.fadeToggle（）。

```
$('.in').click(function () {
    $('#box').fadeIn('slow');
});
$('.out').click(function () {
    $('#box').fadeOut('slow');
});
$('.toggle').click(function () {
    $('#box').fadeToggle();
});
```

上面三个透明度方法只能是从0到100，或者从100到0，如果我们想设置指定值就没有办法了。而JQuery为了解决这个问题提供了.fadeTo（）方法。

```
$('.toggle').click(function () {
    $('#box').fadeTo('slow', 0.33); //0.33 表示值为33
});
```

注意：淡入、淡出效果和显示、隐藏效果一样，具有相同的参数。对于.fadeTo（）方法，如果本身透明度大于指定值，会淡出；否则相反。

四、自定义动画

JQuery提供了几种简单常用的固定动画方便我们使用。但有些时候，这些简单动画无法满足我们更加复杂的需求。这时候，JQuery提供

了一个.animate（）方法来创建我们的自定义动画，满足更多复杂多变的要求。

```
$（'.animate'）.click（function （） {
    $（'#box'）.animate（{
        'width'： '300px',
        'height'： '200px',
        'fontSize'： '50px',
        'opacity'： 0.5
    }）；
}）；
```

注意：一个CSS变化就是一个动画效果。上面的例子中，已经有四个CSS变化，已经实现了多重动画同步运动的效果。

必传的参数只有一个，就是一个键值对CSS变化样式的对象。还有两个可选参数分别为速度和回调函数。

```
$（'.animate'）.click（function （） {
    $（'#box'）.animate（{
        'width'： '300px',
        'height'： '200px'
    }， 1000， function （） {
        alert（'动画执行完毕执行我！'）；
    }）；
}）；
```

到目前位置，我们都是创建的固定位置不动的动画。如果想要实现运动状态的位移动画，那么就必须使用自定义动画，并且结合CSS的绝对定位功能。

```
$（'.animate'）.click（function （） {
    $（'#box'）.animate（{
        'top'： '300px', //先必须设置CSS 绝对定位
        'left'： '200px'
    }）；
}）；
```

在自定义动画中，每次开始运动都必须是初始位置或初始状态，而有时我们想通过当前位置或状态下再进行动画。JQuery提供了自定义动画的累加、累减功能。

```
$（'.animate'）.click（function （） {
```

```
    $('#box').animate({
        'left': '+=100px',
    });
});
```

自定义实现列队动画的方式，有两种：①在回调函数中再执行一个动画；②通过连缀或顺序来实现列队动画。

```
//通过依次顺序实现列队动画
$('.animate').click(function () {
    $('#box').animate({'left': '100px'});
    $('#box').animate({'top': '100px'});
    $('#box').animate({'width': '300px'});
});
```

注意：如果不是同一个元素，就会实现同步动画

```
//通过连缀实现列队动画
$('.animate').click(function () {
    $('#box').animate({
        'left': '100px'
    }).animate({
        'top': '100px'
    }).animate({
        'width': '300px'
    });
});
//通过回调函数实现列队动画
$('.animate').click(function () {
    $('#box').animate({
        'left': '100px'
    }, function () {
        $('#box').animate({
        'top': '100px'
    }, function () {
        $('#box').animate({
            'width': '300px'
        });
        });
    });
});
```

五、列队动画方法

之前我们已经可以实现列队动画了，如果是同一个元素，可以依次

顺序或连缀调用。如果是不同元素，可以使用回调函数。但有时列队动画太多，回调函数的可读性大大降低。为此，JQuery提供了一组专门用于列队动画的方法。

```
//连缀无法实现按顺序列队
$（'#box'）.slideUp（'slow'）.slideDown（'slow'）.css（'backgrou
```

注意：如果动画方法连缀可以实现依次列队，而.css（）方法不是动画方法，会在一开始传入列队之前。那么，可以采用动画方法的回调函数来解决。

```
//使用回调函数，强行将.css（）方法排队到.slideDown（）之后
$（'#box'）.slideUp（'slow'）.slideDown（'slow'， function （）
    $（this）.css（'background'， 'orange'）；
}）；
```

但如果这样的话，当列队动画繁多的时候，可读性不但下降，而原本的动画方法不够清晰。所以，我们的想法是每个操作都是自己独立的方法。那么JQuery提供了一个类似于回调函数的方法：.queue（）。

```
//使用.queue（）方法模拟动画方法跟随动画方法之后
$（'#box'）.slideUp（'slow'）.slideDown（'slow'）.queue（functi
    $（this）.css（'background'， 'orange'）；
}）；
```

现在，我们想继续在.queue（）方法后面再增加一个隐藏动画，这时发现居然无法实现。这是.queue（）特性导致的。有两种方法可以解决这个问题，JQuery的.queue（）的回调函数可以传递一个参数，这个参数是next函数，在结尾处调用这个next（）方法即可再连缀执行列队动画。

```
//使用next 参数来实现继续调用列队动画
$（'#box'）.slideUp（'slow'）.slideDown（'slow'）.queue（functi
    $（this）.css（'background'， 'orange'）；
    next（）；
}）.hide（'slow'）；
```

因为next函数是JQuery1.4版本以后才出现的，而之前我们普遍使用的是.dequeue（）方法。意思为执行下一个元素列队中的函数。

```
//使用.dequeue（）方法执行下一个函数动画
$（'#box'）.slideUp（'slow'）.slideDown（'slow'）.queue（functi
```

```
    $（this）.css（'background'， 'orange'）；
    $（this）.dequeue（）；
}）.hide（'slow'）；
```

如果采用顺序调用，那么使用列队动画方法，就非常清晰了，每一段代表一个列队，而回调函数的嵌套就会杂乱无章。

```
//使用顺序调用的列队，逐个执行，非常清晰
$（'#box'）.slideUp（'slow'）；
$（'#box'）.slideDown（'slow'）；
$（'#box'）.queue（function （） {
    $（this）.css（'background'， 'orange'）；
    $（this）.dequeue（）；
}）
$（'#box'）.hide（'slow'）；
```

.queue（）方法还有一个功能，就是可以得到当前动画列队的长度。当然，这个用法在普通Web开发中用得比较少，这里我们不做详细探讨。

```
//获取当前列队的长度，fx 是默认列队的参数
function count（） {
    return $（"#box"）.queue（'fx'）.length；
}
//在某个动画处调用
$（'#box'）.slideDown（'slow'， function （） {alert（count（）
```

JQuery还提供了一个清理列队的功能方法：.clearQueue（）。把它放入一个列队的回调函数或.queue（）方法里，就可以把剩下为执行的列队给移除。

```
//清理动画列队
$（'#box'）.slideDown（'slow'，function（）{$（this）.clearQueue
```

六、动画相关方法

很多时候需要停止正在运行中的动画，JQuery为此提供了一个.stop（）方法。它有两个可选参数：.stop（clearQueue，gotoEnd）；clearQueue传递一个布尔值，代表是否清空未执行完的动画列队，gotoEnd代表是否直接将正在执行的动画跳转到末状态。

```
//强制停止运行中的
$（'.stop'）.click（function （） {
```

```
    $('#box').stop();
});
//带参数的强制运行
$('.animate').click(function () {
    $('#box').animate({
        'left': '300px'
    }, 1 000);
    $('#box').animate({
        'bottom': '300px'
    }, 1 000);
    $('#box').animate({
        'width': '300px'
    }, 1 000);
    $('#box').animate({
        'height': '300px'
    }, 1 000);
});
$('.stop').click(function () {
    $('#box').stop(true, true);
});
```

注意：第一个参数表示是否取消列队动画，默认为false。如果参数为true，当有列队动画的时候，会取消后面的列队动画。第二参数表示是否到达当前动画结尾，默认为false。如果参数为true，则停止后立即到达末尾处。

有时在执行动画或列队动画时，需要在运动之前有延迟执行，JQuery为此提供了.delay（）方法。这个方法可以在动画之前设置延迟，也可以在列队动画中间加上。

```
//开始延迟1 秒钟，中间延迟1 秒
$('.animate').click(function () {
    $('#box').delay(1 000).animate({
        'left': '300px'
    }, 1 000);
    $('#box').animate({
        'bottom': '300px'
    }, 1 000);
    $('#box').delay(1 000).animate({
        'width': '300px'
    }, 1 000);
    $('#box').animate({
        'height': '300px'
    }, 1 000);
});
```

在选择器的基础章节中，我们提到过一个过滤器：animated。通过这个过滤器可以判断出当前运动的动画是哪个元素。通过这个特点，我们可以避免因为用户快速在某个元素执行动画时，由于动画积累而导致的动画和用户的行为不一致。

```
//递归执行自我，无线循环播放
$（'#box'）.slideToggle（'slow'， function （） {
    $（this）.slideToggle（'slow'， arguments.callee）；
}）；
//停止正在运动的动画，并且设置红色背景
$（'.button'）.click（function（）{
    $（'div：animated'）.stop（）.css（'background'， 'red'）；
}）；
```

七、动画全局属性

JQuery提供了两种全局设置的属性，分别为：$.fx.interval，设置每秒运行的帧数；$.fx.off，关闭页面上所有的动画。

$.fx.interval属性可以调整动画每秒的运行帧数，默认为13毫秒。数字越小越流畅，但可能影响浏览器的性能。

```
//设置运行帧数为1 000毫秒
$.fx.interval=1 000；//默认为13
$（'.button'）.click（function （） {
    $（'#box'）.toggle（3 000）；
}）；
```

$.fx.off属性可以关闭所有动画效果，在非常低端的浏览器，动画可能会出现各种异常问题导致错误。而JQuery设置这个属性，就是用于关闭动画效果的。

```
//设置动画为关闭true
$.fx.off = true；//默认为false
```

补充：在.animate（）方法中，还有一个参数easing。这个参数的大部分参数值需要通过插件来使用。自带的参数有两个：swing（缓动）和linear（匀速），默认为swing。

```
$（'.button'）.click（function （） {
    $（'#box'）.animate（{
        left： '800px'
```

```
    }, 'slow', 'swing');
    $('#pox').animate({
        left: '800px'
    }, 'slow', 'linear');
});
```

第十二章　AJAX

教学要点：

1．AJAX概述；

2．load（）方法；

3．$.get（）和$.post（）；

4．$.getScript（）和$.getJSON（）；

5．$.ajax（）方法；

6．表单序列化。

教学重点：

1．AJAX概述；

2．load（）方法；

3．$.get（）和$.post（）；

4．$.getScript（）和$.getJSON（）；

5．$.ajax（）方法；

6．表单序列化。

教学难点：

理解什么是AJAX及JQuery中的各种AJAX操作的方法。

开篇：AJAX全称为“Asynchronous JavaScript and XML”（异步JavaScript和XML），它并不是JavaScript的一种单一技术，而是利用了一系列交互式网页应用相关的技术所形成的结合体。使用AJAX，我们

可以无刷新状态更新页面，并且实现异步提交，提升了用户体验。

一、**AJAX**概述

AJAX这个概念是由Jesse James Garrett在2005年发明的。它本身不是单一技术，而是一串技术的集合。主要有：

（1）JavaScript，通过用户或其他与浏览器相关事件捕获交互行为。

（2）XMLHttpRequest对象。通过这个对象可以在不中断其他浏览器任务的情况下向服务器发送请求。

（3）服务器上的文件，以XML、HTML或JSON格式保存文本数据。

（4）其他JavaScript，解释来自服务器的数据（比如PHP从MySQL获取的数据）并将其呈现到页面上。

由于AJAX包含众多特性，优势与不足也非常明显。

AJAX优势主要有以下几点：

（1）不需要插件支持（一般浏览器且默认开启JavaScript即可）；

（2）用户体验极佳（不刷新页面即可获取可更新的数据）；

（3）提升Web程序的性能（在传递数据方面做到按需放松，不必整体提交）；

（4）减轻服务器和带宽的负担（将服务器的一些操作转移到客户端）。

AJAX的不足有以下几点：

（1）不同版本的浏览器度XMLHttpRequest对象支持度不足（比如IE5之前）；

（2）前进、后退的功能被破坏（AJAX永远在当前页）；

（3）搜索引擎的支持度不够（搜索引擎还不能理解JS引起变化数据的内容）；

（4）开发调试工具缺乏（相对于其他语言的工具集来说，JS或AJAX调试开发较少）。

异步和同步：

使用AJAX最关键的地方，就是实现异步请求、接受响应及执行回调。那么异步与同步有什么区别呢？我们普通的Web程序开发基本都是同步的，意为执行一段程序才能执行下一段，类似电话中的通话，一个电话接完才能接听下个电话；而异步可以同时执行多条任务，感觉有多条线路，类似于短信，不会因为看一条短信而停止接受另一条短信。AJAX也可以使用同步模式执行，但同步的模式属于阻塞模式，这样会导致多条线路执行时又必须一条一条执行，会让Web页面出现假死状态，所以，一般AJAX大部分采用异步模式。

二、**load**（）方法

JQuery对AJAX做了大量的封装，我们使用起来也较为方便，不需要去考虑浏览器的兼容性。对于封装的方式，JQuery采用了三层封装：最底层的封装方法为$.ajax（）。第二层包括以下三种方法：.load（）、$.get（）和$.post（）。最高层包括$.getScript（）和$.getJSON（）方法。

.load（）方法有三个参数：url（必须，请求html文件的url地址，参数类型为String）、data（可选，发送的key/value数据，参数类型为Object）、callback（可选，成功或失败的回调函数，参数类型为函数Function）。

如果想让AJAX异步载入一段HTML内容，我们只需要一个HTML请求的url即可。

```
//HTML
<input type="button" value="异步获取数据" />
<div id="box"></div>
//JQuery
$（'input'）.click（function （） {
    $（'#box'）.load（'test.html'）;
}）;
```

如果想对载入的HTML进行筛选，那么只要在url参数后面跟着一个选择器即可。

```
//带选择器的url
$（'input'）.click（function （） {
    $（'#box'）.load（'test.html .my'）;
}）;
```

如果是服务器文件，比如.php。一般不仅需要载入数据，还需要向服务器提交数据，那么我们就可以使用第二个可选参数data。向服务器提交数据有两种方式：get和post。

```
//不传递data，则默认get 方式
$（'input'）.click（function （） {
    $（'#box'）.load（'test.php?url=ycku'）;
}）;
//get方式接受的PHP
<?php
    if （$_GET［'url'］ == 'ycku'） {
        echo '瓢城Web 俱乐部官网';
    } else {
        echo '其他网站';
    }
?>
//传递data，则为post 方式
$（'input'）.click（function （） {
    $（'#box'）.load（'test.php', {
        url: 'ycku'
    }）;
}）;
//post方式接受的PHP
<?php
    if （$_POST［'url'］ == 'ycku'） {
        echo '瓢城Web 俱乐部官网';
    } else {
        echo '其他网站';
    }
?>
```

在AJAX数据载入完毕之后，就能执行回调函数callback，也就是第三个参数。回调函数也可以传递三个可选参数：responseText（请求返回）、textStatus（请求状态）、XMLHttpRequest（XMLHttpRequest对象）。

```
$('input').click(function () {
    $('#box').load('test.php', {
        url: 'ycku'
    }, function (response,status, xhr) {
        alert('返回的值为：' + response + '，状态为：' + status +
    });
});
```

注意：status得到的值，如果成功返回数据则为success，否则为error。XMLHttpRequest对象属于JavaScript范畴，可以调用一些属性，见表12-1。

表12-1

属性名	说明
responseText	作为响应主体被返回的文本
responseXML	如果响应主体内容类型是"text/xml"或"application/xml"，则返回包含响应数据的XML DOM文档
status	响应的HTTP状态
statusText	HTTP状态的说明

如果成功返回数据，那么xhr对象的statusText属性则返回'OK'字符串。除了'OK'的状态字符串，statusText属性还提供了一系列其他的值，见表12-2。

表12-2

HTTP状态码	状态字符串	说明
200	OK	服务器成功返回了页面
400	Bad Request	语法错误导致服务器不识别

401	Unauthorized	请求需要用户认证
404	Not found	指定的URL在服务器上找不到
500	Internal Server Error	服务器遇到意外错误，无法完成请求
503	ServiceUnavailable	由于服务器过载或维护导致无法完成请求

三、**$.get**（）和**$.post**（）

.load（）方法是局部方法，因为它需要一个包含元素的JQuery对象作为前缀。而$.get（）和$.post（）是全局方法，无须指定某个元素。对于用途而言，.load（）适合做静态文件的异步获取，而对于需要传递参数到服务器页面的，$.get（）和$.post（）更加合适。

$.get（）方法有四个参数，前面三个参数和.load（）一样，多了一个第四参数type，即服务器返回的内容格式，包括xml、html、script、json、jsonp和text。第一个参数为必选参数，后面三个为可选参数。

```
//使用$.get（）异步返回html 类型
$（'input'）.click（function （） {
    $.get（'test.php'， {
        url： 'ycku'
    }， function （response， status， xhr） {
        if （status == 'success'） {
            $（'#box'）.html（response）；
        }
    }）//type 自动转为html
}）；
```

注意：第四参数type是指定异步返回的类型。一般情况下，type参数是智能判断，并不需要我们主动设置；如果主动设置，则会强行按照指定类型格式返回。

```
//使用$.get（）异步返回xml
$（'input'）.click（function （） {
    $.get（'test.xml'， function （response， status， xhr） {
```

```
        $('#box').html($(response).find('root').find('url
    });//type 自动转为xml
});
```

注意：如果载入的是xml文件，type会智能判断。如果强行设置html类型返回，则会把xml文件当成普通数据全部返回，而不会按照xml格式解析数据。

```
//使用$.get()异步返回json
$.get('test.json', function (response, status, xhr) {
    alert(response[0].url);
});
```

$.post()方法的使用和$.get()基本上一致，它们之间的区别也比较隐晦，基本都是背后的不同，在用户使用上体现不出来。具体区别如下：

（1）GET请求是通过URL提交的，而POST请求则是通过HTTP消息实体提交的；

（2）GET提交有大小限制（2KB），而POST方式不受限制；

（3）GET方式会被缓存下来，可能有安全性问题，而POST没有这个问题；

（4）GET方式通过$_GET[]获取，POST方式通过$_POST[]获取。

```
//使用$.post()异步返回html
$.post('test.php', {
        url: 'ycku'
    }, function (response, status, xhr) {
        $('#box').html(response);
});
```

四、**$.getScript()和$.getJSON()**

JQuery提供了一组用于特定异步加载的方法：$.getScript()用于加载特定的JS文件；$.getJSON()用于专门加载JSON文件。

有时我们希望能够特定的情况再加载JS文件，而不是一开始把所有

的JS文件都加载了，这时课时使用$.getScript（）方法。

```
//点击按钮后再加载JS 文件
$（'input'）.click（function （） {
    $.getScript（'test.js'）;
}）;
$.getJSON（）方法是专门用于加载JSON 文件的，使用方法和之前的类似。
$（'input'）.click（function （） {
    $.getJSON（'test.json', function （response, status, xhr）
        alert（response［0］.url）;
    }）;
}）;
```

五、**$.AJAX**（）

$.AJAX（）是所有AJAX方法中最底层的方法，所有其他方法都是基于$.AJAX（）方法的封装。这个方法只有一个参数，传递一个各个功能键值对的对象。

表12-3 $.AJAX（）方法对象参数表

参数	类型	说明
url	String	发送请求的地址。
type	String	请求方式：POST或GET，默认GET。
timeout	Number	设置请求超时的时间（毫秒）。
data	Object或String	发送到服务器的数据，键值对字符串或对象。
dataType	String	返回的数据类型，比如html、xml、json等。
beforeSend	Function	发送请求前可修改XMLHttpRequest对象的函数。

complete	Function	请求完成后调用的回调函数。
success	Function	请求成功后调用的回调函数。
error	Function	请求失败时调用的回调函数。
global	Boolean	默认为true，表示是否触发全局AJAX。
cache	Boolean	设置浏览器缓存响应，默认为true。如果dataType类型为script或jsonp则为false。
content	DOM	指定某个元素为与这个请求相关的所有回调函数的上下文。
contentType	String	指定请求内容的类型。默认为application/x-www-form-urlencoded。
async	Boolean	是否异步处理。默认为true，false为同步处理。
processData	Boolean	默认为true，数据被处理为URL编码格式。如果为false，则阻止将传入的数据处理为URL编码格式。
dataFilter	Function	用来筛选响应数据的回调函数。
ifModified	Boolean	默认为false，不进行头检测。如果为true，进行头检测，当相应内容与上次请求改变时，请求被认为是成功的。
jsonp	String	指定一个查询参数名称来覆盖默认的jsonp回调参数名callback。

username	String	在HTTP认证请求中使用的用户名。
password	String	在HTTP认证请求中使用的密码。
scriptCharset	String	当远程和本地内容使用不同的字符集时，用来设置script和jsonp请求所使用的字符集。
xhr	Function	用来提供XHR实例自定义实现的回调函数。
traditional	Boolean	默认为false，不使用传统风格的参数序列化。如为true，则使用。

```
//$.AJAX使用
$('input').click(function () {
    $.ajax({
        type: 'POST', //这里可以换成GET
        url: 'test.php',
        data: {
            url: 'ycku'
        },
        success: function (response, stutas, xhr) {
            $('#box').html(response);
        }
    });
});
```

注意：对于data属性，如果是GET模式，可以使用之前的三种形式；如果是POST模式，可以使用之前的两种形式。

六、表单序列化

AJAX用得最多的地方莫过于表单操作，而传统的表单操作是通过submit提交将数据传输到服务器端。如果使用AJAX异步处理，我们需要将每个表单元素逐个获取才能提交。这样工作效率就大大降低了。

```
//常规形式的表单提交
$('form input[type=button]').click(function () {
    $.ajax({
        type: 'POST',
        url: 'test.php',
        data: {
            user: $('form input[name=user]').val(),
            email: $('form input[name=email]').val()
        },
        success: function (response, status, xhr) {
            alert(response);
        }
    });
});
```

使用表单序列化方法.serialize()，会智能地获取指定表单内的所有元素。这样，在面对大量表单元素时，会把表单元素内容序列化为字符串，然后再使用AJAX请求。

```
//使用.serialize()序列化表单的内容
$('form input[type=button]').click(function () {
    $.ajax({
        type: 'POST',
        url: 'test.php',
        data: $('form').serialize(),
        success: function (response, status, xhr) {
            alert(response);
        }
    });
});
```

.serialize()方法不但可以序列化表单内的元素，还可以直接获取单选框、复选框和下拉列表框等内容。

```
//使用序列化得到选中的元素内容
$(': radio').click(function () {
    $('#box').html(decodeURIComponent($(this).serialize()
});
```

除了.serialize()方法外，还有一个可以返回JSON数据的方法：.serializeArray()。这个方法可以直接把数据整合成键值对的JSON对象。

```
$(': radio').click(function () {
    console.log($(this).serializeArray());
```

```
    var json = $（this）.serializeArray（）；
    $（'#box'）.html（json［0］.value）；
});
```

有时，我们可能会在同一个程序中多次调用$.ajax（）方法。而它们很多参数都相同，这个时候我们使用JQuery提供的$.ajaxSetup（）请求默认值来初始化参数。

```
$（'form input［type=button］'）.click（function （） {
    $.ajaxSetup（{
        type： 'POST',
        url： 'test.php',
        data： $（'form'）.serialize（）
    }）；
    $.ajax（{
        success： function （response， status， xhr） {
            alert（response）；
        }
    }）；
}）；
```

在使用data属性传递的时候，如果是以对象形式传递键值对，可以使用$.param（）方法将对象转换为字符串键值对格式。

```
var obj = {a： 1， b： 2， c： 3}；
Var form = $.param（obj）；
alert（form）；
```

注意：使用$.param（）将对象形式的键值对转为URL地址的字符串键值对，可以更加稳定、准确地传递表单的内容。因为有时程序对于复杂的序列化解析能力有限，所以直接传递obj对象要谨慎。

第十三章　AJAX进阶

教学要点：

1．加载请求；

2．错误处理；

3．请求全局事件；

4．JSON和JSONP；

5．jqXHR对象。

教学重点：

1．加载请求；

2．错误处理；

3．请求全局事件；

4．JSON和JSONP；

5．jqXHR对象。

教学难点：

加载请求与错误处理。

开篇：AJAX全称为“Asynchronous JavaScript and XML”（异步JavaScript和XML），它并不是JavaScript的一种单一技术，而是利用了一系列交互式网页应用相关的技术所形成的结合体。使用AJAX，我们可以不刷新状态更新页面，并且实现异步提交，从而提升了用户体验。

一、加载请求

在AJAX异步发送请求时，遇到网速较慢的情况，就会出现请求时间较长的问题。而超过一定时间的请求，用户就会变得不再耐烦而关闭页面。而如果在请求期间能给用户一些提示，比如正在努力加载中...，那么相同的请求时间会让用户体验更加好一些。

JQuery提供了两个全局事件，.ajaxStart（）和.ajaxStop（）。这两个全局事件，只要用户触发了AJAX，请求开始时（未完成其他请求）激活.ajaxStart（），请求结束时（所有请求都结束了）激活.ajaxStop（）。

```
//请求加载提示的显示和隐藏
$（'#box）.ajaxStart（function （） {
    $（this）.show（）；
}）.ajaxStop（function （） {
    $（this）.hide（）；
}）；
```

注意：以上代码在JQuery1.8及以后的版本不再有效，需要使用jquery-migrate向下兼容才能运行。新版本中，必须绑定在document元素上。

```
$（document）.ajaxStart（function （） {
    $（'#box）.show（）；
}）.ajaxStop（function （） {
    $（'#box'）.hide（）；
}）；
//如果请求时间太长，可以设置超时
$.ajax（{
    timeout： 500
}）；
//如果某个ajax不想触发全局时间，可以设置取消
$.ajax（{
    global： true
}）；
```

二、错误处理

AJAX异步提交时，不可能所有情况都是成功完成的，也有因为代码异步文件错误、网络错误导致提交失败的。这时，我们应该把错误报告出来，提醒用户重新提交或提示开发者进行修补。

在之前高层封装中是没有回调错误处理的，比如$.get（）、

$.post（）和.load（）。所以，早期的方法通过全局.ajaxError（）事件方法来返回错误信息。而在JQuery1.5之后，可以通过连缀处理使用局部.error（）方法即可。而对于$.ajax（）方法，不但可以用这两种方法，还有自己的属性方法error：function（）{}。

```
//$.ajax（）使用属性提示错误
$.ajax（{
    type： 'POST',
    url： 'test1.php',
    data： $（'form'）.serialize（），
    success： function （response， status， xhr） {
        $（'#box'）.html（response）；
    },
    error： function （xhr， ） {
        alert（xhr.status + '： ' + xhr.statusText）；
    }
}）；
//$.post（）使用连缀.error（）方法提示错误，连缀方法将被.fail（）取代
$.post（'test1.php'）.error（function （xhr， status， info） {
    alert（xhr.status + '： ' +xhr.statusText）；
    alert（status + '： ' + info）；
}）；
//$.post（）使用全局.ajaxError（）事件提示错误
$（document）.ajaxError（function （event， xhr， settings， int
    alert（xhr.status + '： ' +xhr.statusText）；
    alert（settings+ '： ' + info）；
}）；
```

三、请求全局事件

JQuery对于AJAX操作提供了很多全局事件方法，.ajaxStart（）、.ajaxStop（）、.ajaxError（）等事件方法。它们都属于请求时触发的全局事件。除了这些，还有一些其他全局事件：

```
.ajaxSuccess（），对应一个局部方法：.success（），请求成功完成时执行。
.ajaxComplete（），对应一个局部方法：.complete（），请求完成后注册一个
.ajaxSend（），没有对应的局部方法，只有属性beforeSend，请求发送之前要绑
//$.post（）使用局部方法.success（）
$.post（'test.php'， $（'form'）.serialize（）， function （resp
    $（'#box'）.html（response）；
}）.success（function （response， status， xhr） {
    alert（response）；
}）；
//$.post（）使用全局事件方法.ajaxSuccess（）
$（document）.ajaxSuccess（function （event， xhr， settings） {
```

```
    alert（xhr.responseText）；
    }）；
```

注意：全局事件方法是所有AJAX请求都会触发到，并且只能绑定在document上。而局部方法，则针对某个AJAX。

对于一些全局事件方法的参数，大部分为对象，而这些对象有哪些属性或方法能调用，可以通过遍历方法得到。

```
    //遍历settings对象的属性
    $（document）.ajaxSuccess（function （event， xhr， settings
        for （var i in settings） {
            alert（i）；
        }
    }）；
//$.post（）请求完成的局部方法.complete（）
$.post（'test.php'， $（'form'）.serialize（）， function （resp
    alert（'成功'）；
}）.complete（function （xhr，status） {
    alert（'完成'）；
    }）；
//$.post（）请求完成的全局方法.ajaxComplete（）
$（document）.ajaxComplete（function （event， xhr， settings）
    alert（'完成'）；
}）；

//$.post（）请求发送之前的全局方法.ajaxSend（）
$（document）.ajaxSend（function （event， xhr， settings） {
    alert（'发送请求之前'）；
}）；
//$.ajax（）方法，可以直接通过属性设置即可。
$.ajax（{
    type： 'POST'，
    url： 'test.php'，
    data： $（'form'）.serialize（），
    success： function （response， status， xhr） {
        $（'#box'）.html（response）；
    }，
    complete： function （xhr， status） {
           alert（'完成' + ' - ' + xhr.responseText + ' - ' + s
    }，
    beforeSend： function （xhr， settings） {
        alert（'请求之前' + ' - ' + xhr.readyState + ' - ' + set
    }
}）；
```

注意：在JQuery1.5版本以后，使用.success（）、.error（）和.complete（）连缀的方法，可以用.done（）、.fail（）和.always（）取代。

四、**JSON**和**JSONP**

如果在同一个域下，$.ajax（）方法只要设置dataType属性即可加载JSON文件。而在非同域下，可以使用JSONP，但也是有条件的。

```
//$.ajax（）加载JSON文件
$.ajax（{
    type：  'POST'，
    url：  'test.json'，
    dataType：  'json'，
    success：  function  （response，  status，  xhr）  {
        alert（response［0］.url）；
    }
}）；
```

如果想跨域操作文件的话，我们就必须使用JSONP。JSONP（JSON with Padding）是一个非官方的协议，它允许在服务器端集成Script tags返回至客户端，通过javascript callback的形式实现跨域访问（这仅仅是JSONP简单的实现形式）。

```
//跨域的PHP端文件
<?php
    $arr = array（'a'=>1，'b'=>2，'c'=>3，'d'=>4，'e'=>5）；
    $result = json_encode（$arr）；
    $callback = $_GET［'callback'］；
    echo $callback."（$result）"；
?>
//$.getJSON（）方法跨域获取JSON
$.getJSON（'http：//www.li.cc/test.php?callback=?'，  function
    console.log（response）；
}）；
//$.ajax（）方法跨域获取JSON
$.ajax（{
        url：  'http：//www.li.cc/test.php?callback=?'，
        dataType：  'jsonp'，
        success：  function  （response，  status，  xhr）  {
            console.log（response）；
            alert（response.a）；
        }
    }）；
```

注意：这里的URL如果不想后面跟着?callback=?，那么可以给$.ajax（）方法增加一个属性。

```
//使用jsonp属性
$.ajax（{
    url： 'http：//www.li.cc/test.php'，
    jsonp： 'callback'
}）；
```

五、**jqXHR**对象

在之前，我们使用了局部方法：.success（）、.complete（）和.error（）。这三个局部方法并不是XMLHttpRequest对象调用的，而是$.ajax（）之类的全局方法返回的对象调用的。这个对象，就是jqXHR对象，它是原生对象XHR的一个超集。

```
//获取jqXHR对象，查看属性和方法
var jqXHR = $.ajax（{
    type： 'POST'，
    url： 'test.php'，
    data： $（'form'）.serialize（）
}）；

for （var i in jqXHR） {
    document.write（i + '

'）；
}
```

注意：如果使用jqXHR对象的话，那么建议用.done（）、.always（）和.fail（）代替.success（）、.complete（）和.error（）。在未来版本中，很可能将这三种方法取消。

```
//成功后回调函数
jqXHR.done（function （response） {
    $（'#box'）.html（response）；
}）；
```

使用jqXHR的连缀方式比$.ajax（）的属性方式有三大好处：

（1）可连缀操作，可读性大大提高；

（2）可以多次执行同一个回调函数；

（3）为多个操作指定回调函数。

```
//同时执行多个成功后的回调函数
jqXHR.done（）.done（）；
//多个操作指定回调函数
var jqXHR = $.ajax（'test.php'）；
var jqXHR2 = $.ajax（'test2.php'）；

$.when（jqXHR，  jqXHR2）.done（function （r1，r2） {
    alert（r1［0］）；
    alert（r2［0］）；
}）；
```

第十四章　工具函数

教学要点：

1．字符串操作；

2．数组和对象操作；

3．测试操作；

4．URL操作；

5．浏览器检测；

6．其他操作。

教学重点：

1．字符串操作；

2．数组和对象操作；

3．测试操作；

4．URL操作；

5．浏览器检测；

6．其他操作。

教学难点：

理解什么是工具函数及熟悉使用各种工具。

开篇：工具函数是指直接依附于JQuery对象，针对JQuery对象本身定义的方法，即全局性的函数。它的作用主要是提供比如字符串、数组、对象等操作方面的遍历。

一、字符串操作

在JQuery中，字符串的工具函数只有一个，就是去除字符串左右空格的工具函数：

```
$.trim（）。
//$.trim（）去掉字符串两边空格
var str = ' JQuery';
alert（str）；
alert（$.trim（str））；
```

二、数组和对象操作

JQuery为处理数组和对象提供了一些工具函数，这些函数可以便利地给数组或对象进行遍历、筛选、搜索等操作。

```
//$.each（）遍历数组
var arr = ['张三', '李四', '王五', '马六']；
$.each（arr， function （index， value） {
    $（'#box'）.html（$（'#box'）.html（） + index + '.' + value
}）；
//$.each（）遍历对象
$.each（$.ajax（）， function （name， fn） {
    $（'#box'）.html（$（'#box'）.html（） + name + '.' + '<br /
}）
```

注意：$.each（）中index表示数组元素的编号，默认从0开始。

```
//$.grep（）数据筛选
var arr = [5，2，9，4，11，57，89，1，23，8]；
var arrGrep = $.grep（arr， function （element， index） {
    return element < 6 && index < 5；
}）；
alert（arrGrep）；
```

注意：$.grep（）方法的index是从0开始计算的。

```
//$.map（）修改数据
var arr = [5，2，9，4，11，57，89，1，23，8]；
var arrMap = $.map（arr， function （element， index） {
    if （element < 6 && index < 5） {
        return element + 1；
    }
}）；
```

```
alert（arrMap）；
//$.inArray（）获取查找到元素的下标
var arr = ［5，2，9，4，11，57，89，1，23，8］；
var arrInArray = $.inArray（1， arr）；
alert（arrInArray）；
```

注意：$.inArray（）的下标从0开始计算。

```
//$.merge（）合并两个数组
var arr = ［5，2，9，4，11，57，89，1，23，8］；
var arr2 = ［23，2，89，3，6，7］；
alert（$.merge（arr， arr2））；
//$.unique（）删除重复的DOM 元素
<div></div>
<div></div>
<div class="box"></div>
<div class="box"></div>
<div class="box"></div>
<div></div>
var divs = $（'div'）.get（）；
divs = divs.concat（$（'.box'）.get（））；
alert（$（divs）.size（））；
$.unique（divs）；
alert（$（divs）.size（））；
//.toArray（）合并多个DOM 元素组成数组
alert（$（'li'）.toArray（））；
```

三、测试操作

在JQuery中，数据有着各种类型和状态。有时，我们希望能通过判断数据的类型和状态做相应的操作。JQuery提供了五组测试用的工具函数。

表14-1 测试工具函数

函数名	说明
$.isArray（obj）	判断是否为数组对象，如果是，返回true
$.isFunction（obj）	判断是否为函数，如果是，返回true

$.isEmptyObject（obj）	判断是否为空对象，如果是，返回true
$.isPlainObjet（obj）	判断是否为纯粹对象，如果是，返回true
$.contains（obj）	判断DOM节点是否含有另一个DOM节点，如果是，返回true
$.type（data）	判断数据类型
$.isNumeric（data）	判断数据是否为数值
$.isWindow（data）	判断数据是否为window对象

```
//判断是否为数组对象
var arr = [1，2，3]；
alert（$.isArray（arr））；
//判断是否为函数
var fn = function （） {}；
alert（$.isFunction（fn））；
//判断是否为空对象
var obj = {}
alert（$.isEmptyObject（obj））；
//判断是否由{}或new Object（）创造出的对象
var obj = window；
alert（$.isPlainObject（obj））；
```

注意：如果使用new Object（'name'）；传递参数后，返回类型已不是Object，而是字符串，所以就不是纯粹的原始对象了。

```
//判断第一个DOM 节点是否含有第二个DOM 节点
alert（$.contains（$（'#box'）.get（0），  $（'#pox'）.get（0））
//$.type（）检测数据类型
alert（$.type（window））；
//$.isNumeric检测数据是否为数值
alert（$.isNumeric（5.25））；
//$.isWindow检测数据对象是否为window 对象
alert（$.isWindow（window））；
```

四、**URL**操作

URL操作，在之前的AJAX章节已经讲到过。只有一个方法：$.param（），将对象的键值对转化为URL键值对字符串形式。

```
//$.param（）将对象键值对转换为URL 字符串键值对
var obj = {
name： 'Lee',
age： 100
};
alert（$.param（obj））；
```

五、浏览器检测

由于在早期的浏览器中，分为IE和W3C浏览器。而IE6、IE7、IE8使用的覆盖率还很高，所以，早期的JQuery提供了$.browser工具对象。而现在的JQuery已经废弃删除了这个工具对象，如果还想使用这个对象来获取浏览器版本型号的信息，可以使用兼容插件。

表14-2 $.browser对象属性

属性	说明
webkit	发送请求的地址
mozilla	判断webkit浏览器，如果是，则为true
Number	判断mozilla浏览器，如果是，则为true
safari	判断safari浏览器，如果是，则为true
opera	判断opera浏览器，如果是，则为true
msie	判断IE浏览器，如果是，则为true
version	获取浏览器版本号

//获取火狐浏览器和版本号

alert（$.browser.mozilla + '：' + $.browser.version）；

注意：火狐采用的是mozilla引擎，一般是指火狐；而谷歌Chrome采用的引擎是webkit，一般验证Chrome就用webkit。

还有一种浏览器检测，是对浏览器内容的检测。比如：W3C的透明度为opacity，而IE的透明度为alpha。这个对象是$.support。

表14-3　$.support对象部分属性

属性	说明
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，因为它的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。目前在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。
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中返回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，因为它用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，因为它用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，它会自动插入缺失的tbody。
AJAX	如果浏览器支持AJAX操作，返回true。

```
//$.support.ajax判断是否能创建ajax
alert（$.support.ajax）；
//$.support.opacity设置不同浏览器的透明度
if （$.support.opacity == true） {
   $（'#box'）.css（'opacity'， '0.5'）；
} else {
   $（'#box'）.css（'filter'， 'alpha（opacity=50）'）；
}
```

注意：由于JQuery越来越放弃低端的浏览器，所以检测功能在未来使用频率也越来越低。所以，$.brower已被废弃删除，而$.support.boxModel检测W3C或IE盒子也被删除。并且http://api.jquery.com/JQuery.support/官网也不提供属性列表和解释，给出一个Modernizr第三方小工具来辅组检测。

六、其他操作

JQuery提供了一个预备绑定函数上下文的工具函数：$.proxy（）。这个方法，可以解决诸如外部事件触发调用对象方法时this的指向问题。

```
//$.proxy（）调整this 指向
var obj = {
   name： 'Lee',
   test： function （） {
      alert（this.name）；
   }
};
$（'#box'）.click（obj.test）；//指向的this 为#box 元素
```

```
$（'#box'）.click（$.proxy（obj， 'test'））； //指向的this 为方法
```

第十五章　插件

教学要点：

1．插件概述；

2．验证插件；

3．自动完成插件；

4．自定义插件。

教学重点：

1．插件概述；

2．验证插件；

3．自动完成插件；

4．自定义插件。

教学难点：

理解什么是插件及开发自己想要的功能插件。

开篇：插件（Plugin）也成为JQuery扩展（Extension），是一种遵循一定规范的应用程序接口编写出来的程序。目前JQuery插件已超过几千种，由来自世界各地的开发者共同编写、验证和完善。而对于JQuery开发者而言，直接使用这些插件将快速稳定架构系统、节约项目成本。

一、插件概述

插件是以JQuery的核心代码为基础编写出的符合一定规范的应用程序。也就是说，插件也是JQuery代码，通过js文件引入的方式植入即可。插件的种类很多，主要分为UI类、表单及验证类、输入类、特效类、AJAX类、滑动类、图形图像类、导航类、综合工具类、动画类

等。

引入插件的步骤如下：

（1）必须先引入jquery.js文件，而且在所有插件之前引入；

（2）引入插件；

（3）引入插件的周边，比如皮肤、中文包等。

二、验证插件

Validate.js是JQuery比较优秀的表单验证插件之一。这个插件有两个js文件：一个是主文件，另一个是中文包文件。使用的时候，可以使用min版本。在这里，为了教学，我们未压缩版本。

验证插件包含的两个文件分别为：jquery.validate.js和jquery.validate.messages_zh.js。

```
//HTML内容
<script type="text/javascript" src="jquery.validate.js"></scr
<script type="text/javascript" src="jquery.validate.messages_
<form>
    <p>用户名：<input type="text" class="required" name="userna
    <p>电子邮件：<input type="text" class="required email" name=
    <p>网址：<input type="text" class="url" name="url" />
    <p><input type="submit" value="提交" />
</form>
//JQuery代码
$（function （） {
    $（'form'）.validate（）；
}）；
```

只要通过form元素的JQuery对象调用validate（）方法，就可以实现“必填”“不能小于两位”“电子邮件不正确”“网址不正确”等验证效果。除了js端的validate（）方法调用，表单处也需要相应设置才能最终得到验证效果。

（1）必填项：在表单中设置class="required"。

（2）不得小于两位：在表单设置minlength="2"。

（3）电子邮件：在表单中设置class="email"。

（4）网址：在表单中设置class="url"。

注意：本章就简单地介绍插件的使用，并不针对某个功能的插件进行详细讲解。比如验证插件validate.js，它类似于JQuery，同样具有各种操作方法和功能，需要进行类似手册一样的查询和讲解。所以，我们会在项目中再去详细讲解使用到的插件。

三、自动完成插件

所谓自动完成，就是当用户输入部分字符的时候，智能的搜索出包含字符的全部内容。比如输入a，把匹配的内容列表展示出来。

```
//HTML内容
<script type="text/javascript" src="jquery.autocomplete.js"><
<script type="text/javascript" src="jquery-migrate-1.2.1.js">
<link rel="stylesheet" href="jquery.autocomplete.css" type="t
//JQuery代码
var user= ['about', 'family', 'but', 'black'];
   $('form input[name=username]').autocomplete(user, {
   minChars: 0 //双击显示全部数据
});
```

注意：这个自动完成插件使用的JQuery版本较老，用了一些已被新版本的JQuery废弃删除的方法，这样必须要向下兼容才能有效。所以，去查找插件的时候，要注意一下他坚持的版本。

四、自定义插件

前面我们使用了别人提供好的插件，使用起来非常方便。如果市面上找不到自己满意的插件，并且想自己封装一个插件提供给别人使用。那么就需要自己编写一个JQuery插件了。

按照功能分类，插件的形式可以分为以下三类：

（1）封装对象方法的插件；（也就是基于某个DOM元素的JQuery对象，局部性）

（2）封装全局函数的插件；（全局性的封装）

（3）选择器插件。（类似于.find（））

经过日积月累的插件开发，开发者逐步约定了一些基本要点，以解决各种因为插件导致的冲突、错误等问题。

（1）插件名推荐使用jquery.［插件名］.js，以免和其他js文件或者其他库相冲突；

（2）局部对象附加jquery.fn对象上，全局函数附加在jquery上；

（3）插件内部，this指向是当前的局部对象；

（4）可以通过this.each来遍历所有元素；

（5）所有的方法或插件，必须用分号结尾，避免出现问题；

（6）插件应该返回的是JQuery对象，以保证可链式连缀；

（7）避免插件内部使用$，如果要使用，请传递JQuery进去。

按照以上要点，我们开发一个局部或全局的导航菜单的插件。只要导航的<li>标签内部嵌入要下拉的<ul">，并且class为nav，即可完成下拉菜单。

```
//HTML部分
<ul class="list">
    <li>导航列表
        <ul class="nav">
            <li>导航列表1</li>
            <li>导航列表1</li>
            <li>导航列表1</li>
            <li>导航列表1</li>
            <li>导航列表1</li>
            <li>导航列表1</li>
        </ul>
    </li>
    <li>导航列表
        <ul class="nav">
            <li>导航列表2</li>
            <li>导航列表2</li>
            <li>导航列表2</li>
            <li>导航列表2</li>
            <li>导航列表2</li>
```

```
            <li>导航列表2</li>
        </ul>
    </li>
</ul>
//jquery.nav.js部分
；（function （$） {
    $.fn.extend（{
        'nav'： function （color） {
        $（this）.find（'.nav'）.css（{
                listStyle： 'none',
                margin： 0,
                padding： 0,
                display： 'none',
                color： color
            }）；
            $（this）.find（'.nav'）.parent（）.hover（function （
                $（this）.find（'.nav'）.slideDown（'normal'）；
            }， function （） {
                $（this）.find（'.nav'）.stop（）.slideUp（'normal
            }）；
            return this；
        }
        }）；
    }）（JQuery）；
```

第十六章　知问前端—综合项目

概述

学习要点：

1．项目介绍

2．JQuery UI

3．UI主题

知问系统是一个问答系统。主要功能是：会员提出问题，会员回答问题。目前比较热门的此类网站有知乎http://www.zhihu.com、百度知道http://zhidao.baidu.com/等。这里我们重点参考“知乎”来学习一下它采用的前端效果。

一、项目介绍

我们重点仿照“知乎”的架构模式来搭建界面和布局，以及大部分前端功能。而“百度知道”作为辅助功能来确定我们这个项目需要的前端功能。

从以上知名问答站点中，我们可以确认最主要的前端功能有：①弹出对话框；②前端按钮；③折叠菜单；④选项卡切换；⑤滑动块；⑥日历；⑦自动补全；⑧拖放。

二、**JQuery UI**

JQuery UI是以JQuery为基础的开源JavaScript网页用户界面代码库，包含底层用户交互、动画、特效和可更换主题的可视控件。我们可以直接用它来构建具有很好交互性的web应用程序。

JQuery UI的官网网站为：http://jqueryui.com/。我们下载最新版本的即可。目前本书采用的最新版本为：jquery-ui-1.10.3.custom.zip。里面目录结构如下：

（1）css，包含与JQuery UI相关的CSS文件。

（2）js，包含JQuery UI相关的JavaScript文件。

（3）Development-bundle，包含多个不同的子目录：
demos（JQuery UI示例文件）、docs（JQuery UI的文档文件）、
themes（CSS主题文件）和ui（JQuery ui的JavaScript文件）。

（4）Index.html，可以查看JQuery UI功能的索引页。

三、**CSS**主题

CSS主题就是JQuery UI的皮肤，有各种色调的模版提供使用。对于程序员，可以使用最和网站符合的模版。

我们可以在这里：http://jqueryui.com/themeroller/查看已有模版样式。

创建**header**区

学习要点：

4．创建界面

5．引入UI

一、创建界面

我们首先要设计一个header，这个区域将要设计成永远置顶。也就是，往下拉出滚动条也永远在页面最上层可视区内。在header区，目前先设计LOGO、搜索框、按钮、注册和登录即可。

```
//JS引入和CSS引入
<script type="text/javascript" src="js/jquery.js"></script>
<script type="text/javascript" src="js/jquery.ui.js"></script
<script type="text/javascript" src="js/index.js"></script>
<link rel="shortcut icon" type="image/x-icon" href="img/favic
<link rel="stylesheet" href="css/smoothness/jquery.ui.css" ty
<link rel="stylesheet" href="css/style.css" type="text/css" /
```

```
//HTML
<div id="header">
    <div class="header_main">
        <h1>知问</h1>
        <div class="header_search">
            <input type="text" name="search" class="search" />
        </div>
        <div class="header_button">
            <input type="button" value="查询" id="search_button"
        </div>
        <div class="header_member">
            <a href="###" id="reg_a">注册</a> |
                        <a href="javascript：void（0）" id="login_
        </div>
    </div>
</div>

<div id="reg" title="会员注册">
    表单区
</div>

//CSS
body {
    margin：0；
    padding：0；
    font-size：12px；
    margin：40px 0 0 0；
    background：#fff；
}
#header {
    height：40px；
    width：100%；
    background：url（../img/header_bg.png）；
    position：absolute；
    top：0；
}
#header .header_main {
    width：800px；
    height：40px；
    margin：0 auto；
}
#header .header_main h1 {
    height：40px；
    line-height：40px；
    font-size：20px；
    color：#666；
    margin：0；
```

```
    padding: 0 10px;
    float: left;
}
#header .header_search {
    float: left;
    padding: 6px 0 0 0;
}
#header .header_search .search {
    width: 300px;
    height: 24px;
    border: 1px solid #ccc;
    background: #fff;
    font-size: 14px;
    color: #666;
    text-indent: 5px;
}
#header .header_button {
    float: left;
    padding: 5px;
}
#header .header_member {
    float: right;
    height: 40px;
    line-height: 40px;
}
#header .header_member a {
    font-size: 14px;
    text-decoration: none;
    color: #555555;
}
```

二、引入**UI**

在目前的这个header区域中，有两个地方使用了JQuery UI：一个是button按钮；另一个是dialog对话框。

```
//将button按钮设置成UI
$('#search_button').button();

//将div设置成dialog对话框
$('#reg_a').click(function () {
    $("#reg").dialog();
});
```

对话框**UI**

学习要点：

6. 开启多个dialog

7. 修改dialog样式

8. dialog（）方法的属性

9. dialog（）方法的事件

10. dialog中使用on（）

对话框（dialog）是JQuery UI非常重要的一个功能。它彻底代替了JavaScript的alert（）、prompt（）等方法，也避免了新窗口或页面的繁杂冗余。

一、开启多个**dialog**

我们可以同时打开多个dialog，只要设置不同的id即可实现。

$（'#reg'）.dialog（）；

$（'#login'）.dialog（）；

二、修改**dialog**样式

在弹出的dialog对话框中，在火狐浏览器中打开Firebug或者右击->查看元素。这样，我们可以看看dialog的样式，根据样式进行修改。我们为了和网站主题符合，对dialog的标题背景进行修改。

```
//无需修改ui里的CSS，直接用style.css替代
.ui-widget-header {
   background：url（../img/ui_header_bg.png）；
}
```

注意：其他修改方案类似。

三、**dialog**（）方法的属性

对话框方法有两种形式：①dialog（options）。options是以对象键

值对的形式传参，每个键值对表示一个选项。②dialog（'action'，param）。action是操作对话框方法的字符串，param则是options的某个选项。

表16-1 dialog外观选项

属性	默认值/类型	说明
title	无/字符串	对话框的标题，可以直接设置在DOM元素上
buttons	无/对象	以对象键值对方式，给dialog添加按钮。键是按钮的名称，值是用户点击后调用的回调函数

```
$（'#reg'）.dialog（{
    title： '注册知问',
    buttons： {
        '按钮'： function （） {}
    }
}）；
```

表16-2 dialog页面位置选项

属性	默认值/类型	说明
position	center/字符串	设置一个对话框窗口的坐标位置，默认为center。其他设置值为：left top、top right、bottom left、right bottom（四个角）、top、bottom（顶部或底部，宽度居中）、left或right（左边或右边，高度居中）、center（默认值）。

```
$（'#reg'）.dialog（{
    position： 'left top'
}）；
```

表16-3 dialog大小选项

属性	默认值/类型	说明
width	300/数值	对话框的宽度。默认为300，单位是像素。
height	auto/数值	对话框的高度。默认为auto，单位是像素。
minWidth	150/数值	对话框的最小宽度。默认为150，单位是像素。
minHeight	150/数值	对话框的最小高度。默认为150，单位是像素。
maxWidth	auto/数值	对话框的最大宽度。默认为auto，单位是像素。
maxHeight	auto/数值	对话框的最大高度。默认为auto，单位是像素。

```
$('#reg').dialog({
    height: 500,
    width: 500,
    minWidth: 300,
    minHeight: 300,
    maxWidth: 800,
    maxHeight: 600
});
```

表16-4 dialog视觉选项

属性	默认值/类型	说明
show	false/布尔值	显示对话框时，默认采用淡入效果。

hide	false布尔值	关闭对话框时，默认采用淡出效果。

```
$('#reg').dialog({
    show: true,
    hide: true
});
```

注意：设置true后，默认为淡入淡出，如果想使用别的特效，可以使用以下表格中的字符串参数。

表16-5 show和hide可选特效

特效名称	说明
blind	对话框从顶部显示或消失。
bounce	对话框断断续续地显示或消失，垂直运动。
clip	对话框从中心垂直地显示或消失。
slide	对话框从左边显示或消失。
drop	对话框从左边显示或消失，有透明度变化。
fold	对话框从左上角显示或消失。
highlight	对话框显示或消失，伴随着透明度和背景色的变化。
puff	对话框从中心开始缩放。显示时“收缩”，消失时“生长”。
scale	对话框从中心开始缩放。显示时“生长”，消失时“收缩”。
pulsate	对话框以闪烁形式显示或消失。

```
$('#reg').dialog({
    show: 'blind',
    hide: 'blind'
});
```

表16-6 dialog行为选项

属性	默认值/类型	说明
autoOpen	true/布尔值	默认为true，调用dialog（）方法时就会打开对话框；如果为false，对话框不可见，但对话框已创建，可以通过dialog（'open'）才能可见。
draggable	true/布尔值	默认为true，可以移动对话框，false无法移动。
resizable	True/布尔值	默认为true，可以调整对话框大小，false无法调整。
modal	false/布尔值	默认为false，对话框外可操作，true对话框会遮罩一层灰纱，无法操作。
closeText	无/字符串	设置关闭按钮的title文字。

```
$('#reg').dialog({
    autoOpen: false,
    draggable: false,
    resizable: false,
    modal: true,
    closeText: '关闭'
});
```

四、**dialog**（）方法的事件

除了属性设置外，dialog（）方法也提供了大量的事件。这些事件可以给各种不同状态时提供回调函数。这些回调函数中的this值等于对

话框内容的div对象，不是整个对话框的div。

表16-7 dialog事件选项

事件名	说明
focus	当对话框被激活时（首次显示以及每次在上面点击），会调用focus方法。该方法有两个参数（event，ui）。此事件中的ui参数为空。
create	当对话框被创建时会调用create方法，该方法有两个参数（event，ui）。此事件中的ui参数为空。
open	当对话框被显示时（首次显示或调用dialog（'open'）方法），会调用open方法。该方法有两个参数（event，ui）。此事件中的ui参数为空。
beforeClose	当对话框将要关闭时（当单击关闭按钮或调用dialog（'close'）方法），会调用beforeclose方法。如果该函数返回false，对话框将不会被关闭。关闭的对话框可以用dialog（'open'）重新打开。该方法有两个参数（event，ui）。此事件中的ui参数为空。
close	当对话框将要关闭时（当单击关闭按钮或调用dialog（'close'）方法），会调用close方法。关闭的对话框可以用dialog（'open'）重新打开。该方法有两个参数（event，ui）。此事件中的ui参数为空。
drag	当对话框移动时，每次移动一点均会调用drag方法。该方法有两个参数。该方法有两个参数（event，ui）。此事件中的ui有两个属性对象： position，得到当前移动的坐标，有两个子属

	性：top和left。 offset，得到当前移动的坐标，有两个子属性：top和left。
dragStart	当开始移动对话框时，会调用dragStart方法。该方法有两个参数（event，ui）。此事件中的ui有两个属性对象： ①position，得到当前移动的坐标，有两个子属性：top和left。 ②offset，得到当前移动的坐标，有两个子属性：top和left。
dragStop	当开始移动对话框时，会调用dragStop方法。该方法有两个参数（event，ui）。此事件中的ui有两个属性对象： ①position，得到当前移动的坐标，有两个子属性：top和left。 ②offset，得到当前移动的坐标，有两个子属性：top和left。
resize	当对话框拉升大小的时候，每一次拖拉都会调用resize方法。该方法有两个参数（event，ui）。此事件中的ui有四个属性对象： ①size，得到对话框的大小有两个子属性：width和height。 ②position，得到对话框的坐标有两个子属性：top和left。 ③originalSize，得到对话框原始的大小有两个子属性：width和height。 ④originalPosition，得到对话框原始的坐标有两个子属性：top和left。
	当开始拖拉对话框时，会调用resizeStart方法。该方法有两个参数（event，ui）。此事件中的ui有四个属性对象： ①size，得到对话框的大小有两个子属性：width和height。

resizeStart	②position，得到对话框的坐标有两个子属性：top和left。 ③originalSize，得到对话框原始的大小有两个子属性：width和height。 ④originalPosition，得到对话框原始的坐标有两个子属性：top和left。
resizeStop	当结束拖拉对话框时，会调用resizeStart方法。该方法有两个参数（event，ui）。此事件中的ui有四个属性对象： ①size，得到对话框的大小有两个子属性：width和height。 ②position，得到对话框的坐标有两个子属性：top和left。 ③originalSize，得到对话框原始的大小有两个子属性：width和height。 ④originalPosition，得到对话框原始的坐标有两个子属性：top和left。

```
//当对话框获得焦点后
$（'#reg'）.dialog（{
    focus： function （e， ui） {
        alert（'获得焦点'）；
    }
}）；

//当创建对话框时
$（'#reg'）.dialog（{
    create： function （e， ui） {
        alert（'创建对话框'）；
    }
}）；

//当将要关闭时
$（'#reg'）.dialog（{
    beforeClose： function （e， ui） {
        alert（'关闭前做的事！'）；
        return flag；
    }
}）；

//关闭对话框时
```

```
$（'#reg'）.dialog（{
    close： function （e， ui） {
        alert（'关闭！'）；
    }
}）；

//对话框移动时
$（'#reg'）.dialog（{
    drag： function （e， ui） {
        alert（'top：' + ui.position.top + '\\n'
            + 'left：' + ui.position.left）；
    }
}）；

//对话框开始移动时
$（'#reg'）.dialog（{
    dragStart： function （e， ui） {
        alert（'top：' + ui.position.top + '\\n'
            + 'left：' + ui.position.left）；
    }
}）；

//对话框结束移动时
$（'#reg'）.dialog（{
    dragStop： function （e， ui） {
        alert（'top：' + ui.position.top + '\\n'
            + 'left：' + ui.position.left）；
    }
}）；

//调整对话框大小时
$（'#reg'）.dialog（{
    resize： function （e， ui） {
        alert（'size：' + ui.size.width + '\\n'
            + 'originalSize：' + ui.originalSize.width）；
}
}）；

//开始调整对话框大小时
$（'#reg'）.dialog（{
    resizeStart： function （e， ui） {
        alert（'size：' + ui.size.width + '\\n'
            + 'originalSize：' + ui.originalSize.width）；
    }
}）；

//结束调整对话框大小时
$（'#reg'）.dialog（{
```

```
    resizeStop: function (e, ui) {
        alert('size: ' + ui.size.width + '\\n'
            + 'originalSize: ' + ui.originalSize.width);
    }
});
```

表16-8 dialog('action', param)方法

方法	返回值	说明
dialog('open')	JQuery对象	打开对话框
dialog('close')	JQuery对象	关闭对话框
dialog('destroy')	JQuery对象	删除对话框，直接阻断了dialog
dialog('isOpen')	布尔值	判断对话框是否打开状态
dialog('widget')	JQuery对象	获取对话框的JQuery对象
dialog('option', param)	一般值	获取options属性的值
dialog('option', param, value)	JQuery对象	设置options属性的值

```
//初始隐藏对话框
$('#reg').dialog({
    autoOpen: false
});

//打开对话框
$('#reg_a').click(function () {
    $('#reg').dialog('open');
});

//关闭对话框
$('#reg').click(function () {
    $('#reg').dialog('close');
});
```

```
//判断对话框处于打开或关闭状态
$（document）.click（function （） {
    alert（$（'#reg'）.dialog（'isOpen'））；
}）；

//将指定对话框置前
$（document）.click（function （） {
    $（'#reg'）.dialog（'moveToTop'）；
}）；

//获取某个options的param选项的值
var title = $（'#reg'）.dialog（'option'， 'title'）；
alert（title）；

//设置某个options的param选项的值
$（'#reg'）.dialog（'option'， 'title'， '注册知问'）；
```

五、**dialog**中使用**on**（）

在dialog的事件中，提供了使用on（）方法处理的事件方法。

表16-9 on（）方法触发的对话框事件

特效名称	说明
dialogfocus	得到焦点时触发
dialogopen	显示时触发
dialogbeforeclose	将要关闭时触发
dialogclose	关闭时触发
dialogdrag	每一次移动时触发
dialogdragstart	开始移动时触发
dialogdragstop	移动结束后触发
dialogresize	每次调整大小时触发

dialogresizestart	开始调整大小时触发
dialogresizestop	结束调整大小时触发

```
$（'#reg'）.on（'dialogclose', function （） {
    alert（'关闭'）;
}）;
```

按钮**UI**

学习要点：

11．使用button按钮

12．修改button样式

13．button（）方法的属性

14．button（'action', param）

15．单选、复选按钮

按钮（button）可以给生硬的原生按钮或者文本提供更多丰富多彩的外观。它不仅可以设置按钮或文本，还可以设置单选按钮和多选按钮。

一、使用**button**按钮

使用button按钮UI的时候，不一定必须是input按钮形式，普通的文本也可以设置成button按钮。

$（'#search_button'）.button（）；

二、修改**button**样式

在弹出的button对话框中，在火狐浏览器中打开Firebug或者右击->查看元素。这样，我们可以看看button的样式，根据样式进行修改。我们为了和网站主题符合，对dialog的标题背景进行修改。

```
//无需修改ui里的CSS，直接用style.css替代
.ui-state-default，  .ui-widget-content .ui-state-default，  .ui
   background：url（../img/ui_header_bg.png）；
}
.ui-state-active，  .ui-widget-content .ui-state-active，  .ui-w
   background：url（../img/ui_white.png）；
}
```

注意：其他修改方案类似。

三、**button**（）方法的属性

按钮方法有两种形式：①button（options）。options是以对象键值对的形式传参，每个键值对表示一个选项。②button（'action'，param）。action是操作对话框方法的字符串，param则是options的某个选项。

表16-10　Button按钮选项

属性	默认值/类型	说明
disabled	false/布尔值	默认为false，设置为true时，按钮是非激活的。
label	无/字符串	对应按钮上的文字。如果没有，HTML内容将被作为按钮的文字。

icons	无/字符串	对应按钮上的图标。在按钮文字前面和后面都可以放置一个图标，通过对象键值对的方式完成：
		{
		primary：'ui-icon-search'，
		secondary：'ui-icon-search'
		}

text	true/布尔值	当时设置为false时，不会显示文字，但必须指定一个图标。

```
$('#search_button').button({
    disabled: false,
    icons: {
        primary: 'ui-icon-search',
    },
    label: '查找',
    text: false,
});
```

注意：对于button的事件方法，只有一个create，当创建button时调用。

四、**button（'action'，param）**

button（'action'，param）方法能设置和获取按钮。action表示指定操作的方式。

表16-11 dialog（'action', param）方法

方法	返回值	说明
button（'disable'）	JQuery对象	禁用按钮
button（'enable'）	JQuery对象	启用按钮
button（'destroy'）	JQuery对象	删除按钮，直接阻断了button
button（'refresh'）	JQuery对象	更新按钮布局
button（'widget'）	JQuery对象	获取对话框的JQuery对象
button（'option'，param）	一般值	获取options属性的值
button（'option'，param，value）	JQuery对象	设置options属性的值

```
//禁用按钮
$（'#search_button'）.button（'disable'）；

//启用按钮
$（'#search_button'）.button（'enable'）；

//删除按钮
$（'#search_button'）.button（'destroy'）；

//更新按钮，刷新按钮
$（'#search_button'）.button（'refresh'）；

//得到button的JQuery对象
$（'#search_button'）.button（'widget'）；

//得到button的options值
alert（$（'#search_button'）.button（'option'， 'label'））；

//设置button的options值
$（'#search_button'）.button（'option'， 'label'， '搜索'）；
```

注意：对于UI上自带的按钮，比如dialog上的，我们可以通过Firebug查找得到JQuery对象。

```
$（'#reg'）.parent（）.find（'button'）.eq（1）.button（'disable
```

五、单选框、复选框

button按钮不但可以设置成普通的按钮，对于单选框、复选框同样有效。

```
//HTML单选框
<input type="radio" name="sex" value="male" id="male">
   <label for="male">男</label>
</input>
<input type="radio" name="sex" value="female" id="female">
   <label for="female">女</label>
</input>

//JQuery单选框
$（'#reg input［type=radio］'）.button（）；

//JQuery单选框改
$（'#reg'）.buttonset（）；//HTML部分做成一行即可

//HTML复选框
```

```
<input type="checkbox" name="color" value="red" id="red">
    <label for="red">红</label>
</input>
<input type="checkbox" name="color" value="green" id="green">
    <label for="green">绿</label>
</input>
<input type="checkbox" name="color" value="yellow" id="yellow
    <label for="yellow">黄</label></input>
<input type="checkbox" name="color" value="orange" id="orange
    <label for="orange">橙</label>
</input>
//JQuery复选框
$（'#reg input［type=radio］'）.button（）；

//JQuery复选框改
$（'#reg'）.buttonset（）；
```

创建注册表单

学习要点：

16. HTML部分

17. CSS部分

18. JQuery部分

通过前面已学的JQuery UI部件，我们来创建一个注册表单。

一、**HTML**部分

```
<div id="reg" title="会员注册">
    <p>
        <label for="user">账号：</label>
        <input type="text" name="user" class="text" id="user" t
        <span class="star">*</span>

    <p>
        <label for="pass">密码：</label>
        <input type="text" name="pass" class="text" id="pass" t
        <span class="star">*</span>

    <p>
        <label for="email">邮箱：</label>
```

```
        <input type="text" name="email" class="text" id="email"
        <span class="star">*</span>

    <p>
        <label>性别：</label>
        <input type="radio" name="sex" id="male" checked="check

    <p>
        <label for="date">生日：</label>
        <input type="text" name="date" readonly="readonly" clas

</div>
```

二、CSS部分

```
#reg {
    padding：15px；
}
#reg p {
    margin：10px 0；
    padding：0；
}
#reg p label {
    font-size：14px；
    color：#666；
}
#reg p .star {
    color：red；
}
#reg .text {
    border-radius：4px；
    border：1px solid #ccc；
    background：#fff；
    height：25px；
    width：200px；
    text-indent：5px；
    color：#666；
}
```

三、JQuery部分

```
$（'#reg'）.dialog（{
    autoOpen：  true，
    modal：  true，
    resizable：  false，
    width：  320，
    height：  340，
```

```
        buttons: {
            '提交': function () {}
        },
    });

    $('#reg').buttonset();
    $('#date').datepicker();
    $('#reg input[title]').tooltip();
```

工具提示**UI**

学习要点：

19. 调用tooltip（）方法

20. 修改tooltip（）样式

21. tooltip（）方法的属性

22. tooltip（）方法的事件

23. tooltip（）中使用on（）

工具提示（tooltip）是一个非常实用的UI。它彻底扩展了HTML中的title属性，让提示更加丰富和可控制，全面提升了用户体验。

一、调用**tooltip**（）方法

在调用tooltip（）方法之前，首先需要针对元素设置相应的title属性。

<input type="text" name="user" class="text" id="user" title="请输入账号，不小于2位！" />

$（'#user'）.tooltip（）；

二、修改**tooltip**（）样式

在弹出的tooltip提示框后，在火狐浏览器中打开Firebug或者右击->查看元素。这样，我们可以看看tooltip的样式，根据样式进行修改。

```
//无需修改ui里的CSS，直接用style.css替代
.ui-tooltip {
    color：red；
}
```

注意：其他修改方案类似。

三、**tooltip**（）方法的属性

工具提示方法有两种形式：①tooltip（options）。options是以对象键值对的形式传参，每个键值对表示一个选项。②tooltip（'action'，param）。action是操作对话框方法的字符串，param则是options的某个选项。

表16-12 tooltip外观选项

属性	默认值/类型	说明
disabled	false/布尔值	设置为true，将禁止显示工具提示
content	无/字符串	设置title内容
items	无/字符串	设置选择器以限定范围
tooltipClass	无/字符串	引入class形式的CSS样式

```
$（'［title］'）.tooltip（{
    disabled： false，
    content： '改变文字'，
    items： 'input'，
    tooltipClass： 'reg_tooltip'
}）；
```

表16-13 tooltip页面位置选项

属性	默认值/类型	说明
		使用对象的键值对赋值，有两个

position	无/对象	属性：my和at表示坐标。my是以目标点左下角为基准，at以my为基准。

```
$('#user').tooltip({
    position: {
        my: 'left center',
        at: 'right+5 center'
    }
});
```

表16-14 tooltip视觉选项

属性	默认值/类型	说明
show	false/布尔值	显示对话框时，默认采用淡入效果
hide	false布尔值	关闭对话框时，默认采用淡出效果

```
$('#user').tooltip({
    show: false,
    hide: false,
});
```

注意：设置true后，默认为淡入或淡出，如果想使用别的特效，可以使用以下表格中的字符串参数。

表16-15 show和hide可选特效

特效名称	说明
blind	工具提示从顶部显示或消失
bounce	工具提示断断续续地显示或消失，垂直运动
clip	工具提示从中心垂直地显示或消失

slide	工具提示从左边显示或消失
drop	工具提示从左边显示或消失，有透明度变化
fold	工具提示从左上角显示或消失
highlight	工具提示显示或消失，伴随着透明度和背景色的变化
puff	工具提示从中心开始缩放。显示时“收缩”，消失时“生长”
scale	工具提示从中心开始缩放。显示时“生长”，消失时“收缩”
pulsate	工具提示以闪烁形式显示或消失

```
$('#user').tooltip({
    show:  'blind',
    hide:  'blind',
});
```

表16-16 tooltip行为选项

属性	默认值/类型	说明
track	false/布尔值	设置为true，能跟随鼠标移动

```
$('#user').tooltip({
    track:  true,
});
```

四、**tooltip**（）方法的事件

除了属性设置外，tooltip（）方法也提供了大量的事件。这些事件可以给各种不同状态时提供回调函数。这些回调函数中的this值等于对话框内容的div对象，不是整个对话框的div。

表16-17 tootip事件选项

事件名	说明
create	当工具提示被创建时，会调用create方法。该方法有两个参数（event，ui）。此事件中的ui参数为空。
open	当工具提示被显示时，会调用open方法。该方法有两个参数（event，ui）。此事件中的ui有一个参数tooltip，返回是工具提示的JQuery对象。
close	当工具提示关闭时，会调用close方法。关闭的工具提示可以用tooltip（'open'）重新打开，该方法有两个参数（event，ui）。此事件中的ui有一个参数tooltip，返回是工具提示的JQuery对象。

```
//当创建工具提示时
$('#user').tooltip({
    create: function () {
        alert('创建触发!');
    }
});

//当工具提示关闭时
$('#user').tooltip({
    close: function () {
        alert('关闭触发');
    }
});

//当工具提示打开时
$('#user').tooltip({
    open: function () {
        alert('打开触发');
    }
});
```

表16-18 tooltip（'action', param）方法

方法	返回值	说明
tooltip（'open'）	JQuery对象	打开工具提示。
tooltip（'close'）	JQuery对象	关闭工具提示。
tooltip（'disable'）	JQuery对象	禁用工具提示。
tooltip（'enable'）	JQuery对象	启用工具提示。
tooltip（'destroy'）	JQuery对象	删除工具提示，直接阻断了tooltip。
tooltip（'widget'）	JQuery对象	获取工具提示的JQuery对象。
tooltip（'option'，param）	一般值	获取options属性的值。
tooltip（'option'，param，value）	JQuery对象	设置options属性的值。

```
//打开工具提示
$（'#user'）.tooltip（'open'）；

//关闭工具提示
$（'#user'）.tooltip（'close'）；

//禁用工具提示
$（'#user'）.tooltip（'disable'）；

//启用工具提示
$（'#user'）.tooltip（'enable'）；

//删除工具提示
$（'#user'）.tooltip（'destroy'）；

//获取工具提示JQuery对象
$（'#user'）.tooltip（'widget'）；

//获取某个options的param选项的值
```

```
var title = $('#user').tooltip('option', 'content');
alert(title);

//设置某个options的param选项的值
$('#reg').dialog('option', 'content', '提示内容');
```

五、**tooltip**（）中使用**on**（）

在tooltip的事件中，提供了使用on（）方法处理的事件方法。

表16-19 on（）方法触发的对话框事件

特效名称	说明
tooltipopen	显示时触发。
tooltipclose	每一次移动时触发。

```
$('#reg').on('tooltipopen', function () {
    alert('打开时触发!');
});
```

自动补全**UI**

学习要点：

24. 调用autocomplete（）方法

25. 修改autocomplete（）样式

26. autocomplete（）方法的属性

27. autocomplete（）方法的事件

28. autocomplete中使用on（）

自动补全（autocomplete）是一个可以减少用户输入完整信息的UI工具。一般先输入邮箱、搜索关键字等，然后提取出相应完整字符串供用户选择。

一、调用**autocomplete**（）方法

```
$（'#email'）.autocomplete（{
    source：  ［'aaa@163.com'，  'bbb@163.com'，  'ccc@163.com'］，
}）；
```

二、修改**autocomplete**（）样式

由于autocomplete（）方法是弹出窗口，然后是鼠标悬停的样式。我们通过Firebug想获取到悬停时背景的样式，可以直接通过jquery.ui.css找到相应的CSS。

```
//无需修改ui里的CSS，直接用style.css替代
.ui-menu-item a.ui-state-focus {
    background：url（../img/ui_header_bg.png）；
}
```

注意：其他修改方案类似。

三、**autocomplete**（）方法的属性

自动补全方法有两种形式：①autocomplete（options）。options是以对象键值对的形式传参，每个键值对表示一个选项。
②autocomplete（'action'，param）。action是操作对话框方法的字符串，param则是options的某个选项。

表16-20 autocomplete外观选项

属性	默认值/类型	说明
disabled	false/布尔值	设置为true，将禁止显示自动补全。
source	无/数组	指定数据源，可以是本地的，也可以是远程的。
minLength	1/数值	默认为1，触发补全列表最少输入字符数。

delay	300/数值	默认为300毫秒，延迟显示设置。
autoFocus	false/布尔值	设置为true时，第一个项目会自动被选定。

```
$('#email').autocomplete({
    source: ['aaa@163.com', 'bbb@163.com', 'ccc@163.com'],
    disabled: false,
    minLength: 2,
    delay: 50,
    autoFocus: true,
});
```

表16-21 autocomplete页面位置选项

属性	默认值/类型	说明
position	无/对象	使用对象的键值对赋值，有两个属性：my和at表示坐标。my以目标点左上角为基准，at以目标点右下角为基准。

```
$('#email').autocomplete({
    position: {
        my: 'left center',
        at: 'right center'
    }
});
```

三、**autocomplete**（）方法的事件

除了属性设置外，autocomplete（）方法也提供了大量的事件。这些事件可以给各种不同状态时提供回调函数。这些回调函数中的this值等于对话框内容的div对象，不是整个对话框的div。

表16-22 autocomplete事件选项

事件名	说明

create	当自动补全被创建时，会调用create方法。该方法有两个参数（event，ui）。此事件中的ui参数为空。
open	当自动补全被显示时，会调用open方法。该方法有两个参数（event，ui）。此事件中的ui参数为空。
close	当自动补全被关闭时，会调用close方法。该方法有两个参数（event，ui）。此事件中的ui参数为空。
focus	当自动补全获取焦点时，会调用focus方法。该方法有两个参数（event，ui）。此事件中的ui有一个子属性对象item，分别有两个属性：label，补全列表显示的文本；value，将要输入框的值。一般label和value值相同。
select	当自动补全获被选定时，会调用select方法。该方法有两个参数（event，ui）。此事件中的ui有一个子属性对象item，分别有两个属性：label，补全列表显示的文本；value，将要输入框的值。一般label和value值相同。
change	当自动补全失去焦点且内容发生改变时，会调用change方法。该方法有两个参数（event，ui）。此事件中的ui参数为空。
search	当自动补全搜索完成后。会调用search方法。该方法有两个参数（event，ui）。此事件中的ui参数为空。
response	当自动补全搜索完成后，在菜单显示之前，会调用response方法。该方法有两个参数（event，ui）。此事件中的ui参数有一个子对象content，它会返回label和value值，可通过遍

	历了解。

```
$（'#email'）.autocomplete（{
    source： ［'aaa@163.com'， 'bbb@163.com'， 'ccc@163.com'］，
    disabled： false，
    minLength： 1，
    delay： 0，
    focus： function （e， ui） {
       ui.item.value = '123'；
    }，
    select： function （e， ui） {
       ui.item.value = '123'；
    }，
    change： function （e， ui） {
       alert（''）；
    }，
    search： function （e， ui） {
       alert（''）；
    }，
}）；
```

表16-23 autocomplete（'action', param）方法

方法	返回值	说明
autocomplete（'close'）	JQuery对象	关闭自动补齐。
autocomplete（'disable'）	JQuery对象	禁用自动补齐。
autocomplete（'enable'）	JQuery对象	启用自动补齐。
autocomplete（'destroy'）	JQuery对象	删除自动补齐，直接阻断。
autocomplete（'widget'）	JQuery对象	获取工具提示的JQuery对象。
autocomplete（'search'，value）	JQuery对象	在数据源获取匹配的字符串。

autocomplete（'option'，param）	一般值	获取options属性的值。
autocomplete（'option'，param，value）	JQuery对象	设置options属性的值。

```
//关闭自动补全
$（'#email'）.autocomplete（'close'）；

//禁用自动补全
$（'#email'）.autocomplete（'disable'）；

//启用自动补全
$（'#email'）.autocomplete（'enable'）；

//删除自动补全
$（'#email'）.autocomplete（'destroy'）；

//获取自动补全JQuery对象
$（'#email'）.autocomplete（'widget'）；

//设置自动补全search
$（'#email'）.autocomplete（'search'， ''）；

//获取某个options的param选项的值
var delay = $（'#email'）.autocomplete（'option'， 'delay'）；
alert（delay）；

//设置某个options的param选项的值
$（'#email'）.dialog（'option'， 'delay'， 0）；
```

四、**autocomplete**中使用**on**（）

在autocomplete的事件中，提供了使用on（）方法处理的事件方法。

表16-24 on（）方法触发的对话框事件

事件名称	说明
autocompleteopen	显示时触发。

autocompleteclose	关闭时触发。
autocompletesearch	查找时触发。
autocompletefocus	获得焦点时触发。
autocompleteselect	选定时触发。
autocompletechange	改变时触发。
autocompleteresponse	搜索完毕后，显示之前。

```
$('#reg').on('autocompleteopen', function () {
    alert('打开时触发!');
});
```

邮箱自动补全

学习要点：

29. 数据源function

30. 邮箱自动补全

本节课，我们通过自动补全source属性的function回调函数，来动态地设置我们的数据源，以达到可以实现邮箱补全的功能。

一、数据源**function**

自动补全UI的source不但可以是数组，也可以是function回调函数。下面提供了自带的两个参数设置动态的数据源。

```
$('#email').autocomplete({
    source: function (request, response) {
        alert(request.term);                    //可以获取你输入
        response(['aa', 'aaaa', 'aaaaaa', 'bb']); //展示补全
    },
});
```

注意：这里的response不会根据你搜索关键字而过滤无关结果，而是把整个结果全部呈现出来。因为source数据源，本身就是给你动态改变的，由你自定义，从而放弃系统内置的搜索能力。

二、邮箱自动补全

```
$('#email').autocomplete({
    autoFocus: true,
    delay: 0,
    source: function (request, response) {
        var hosts = ['qq.com', '163.com', '263.com', 'gmail.c
            term = request.term,          //获取输入值
            ix = term.indexOf('@'), //@
            name = term, //用户名
            host = '', //域名
            result = []; //结果

        //结果第一条是自己输入
        result.push(term);

        if (ix > -1) {//如果有@的时候
            name = term.slice(0, ix); //得到用户名
            host = term.slice(ix + 1); //得到域名
        }

        if (name) {
                //得到找到的域名
            var findedHosts = (host ? $.grep(hosts, function
                    return value.indexOf(host) > -1;
                }): hosts),
                //最终列表的邮箱
                findedResults = $.map(findedHosts, function (v
                    return name + '@' + value;
                });

            //增加一个自我输入
            result = result.concat(findedResults);
        }
        response(result);
    },
});
```

日历**UI**

学习要点：

31. 调用datepicker（）方法

32. 修改datepicker（）样式

33. datepicker（）方法的属性

34. datepicker（）方法的事件

日历（datepicker）UI可以让用户更加直观、方便的输入日期，并且还考虑了不同国家的语言限制，包括汉语。

一、调用**datepicker**（）方法

$（'#date'）.datepicker（）；

二、修改**datepicker**（）样式

日历UI的header背景和对话框UI的背景采用的是同一个class，所以，在此之前已经被修改。所以，这里无需再修改了。

```
//无需修改ui里的CSS，直接用style.css替代
.ui-widget-header {
   background：url（../img/ui_header_bg.png）；
}

//修改当天日期的样式
.ui-datepicker-today .ui-state-highlight {
   border：1px solid #eee；
   color：#f60；
}

//修改选定日期的样式
.ui-datepicker-current-day .ui-state-active {
   border：1px solid #eee；
   color：#06f；
}
```

注意：其他修改方案类似。

三、**datepicker**（）方法的属性

日历方法有两种形式：①datepicker（options）。options是以对象键值对的形式传参，每个键值对表示一个选项。②datepicker（'action'，param）。action是操作对话框方法的字符串，param则是options的某个选项。

表16-25 datepicker国际化选项

属性	默认值/类型	说明
dateFormat	mm/dd/yy/时间	指定日历返回的日期格式。
dayNames	英文日期/数组	以数组形式指定星期中的天的长格式。比如Sunday、Monday等。中文：星期日。
dayNamesShort	英文日期/数组	以数组形式指定星期中的天的短格式。比如Sun、Mon等。
dayNamesMin	英文日期/数组	以数组形式指定星期中的天的最小格式。比如Su、Mo等。
monthNames	英文月份/数组	以数组形式指定月份的长格式名称（January、February等）。数组必须从January开始。
monthNamesShort	英文月份/数组	以数组形式指定月份的短格式名称（Jan、Feb等）。数组必须从January开始。
altField	无/字符串	为日期选择器指定一个<input>域。
altFormat	无/字符串	添加到<input>域的可选日期格式。

appendText	无/字符串	在日期选择器的<input>域后面附加文本。
showWeek	false/布尔值	显示周。
weekHeader	'Wk'/字符串	显示周的标题。
firstDay	0/数值	指定日历中的星期从星期几开始。0表示星期日。

注意：在默认情况下，日历显示为英文。如果你想使用中文日历，直接引入中文语言包即可。或者把中文语言包的几行代码整合到某个js文件里即可。

表16-26　日期格式代码

代码	说明
d	月份中的天，从1到31。
dd	月份中的天，从01到31。
o	年份中的天，从1到366。
oo	年份中的天，从001到366。
D	星期中的天的缩写名称（Mon、Tue等）。
DD	星期中的天的全写名称（Monday、Tuesday等）。
m	月份，从1到12。
mm	月份，从01到12。
M	月份的缩写名称（Jan、February等）。

MM	月份的全写名称（January、February等）。
y	两位数字的年份（14表示2014）。
yy	四位数字的年份（2014）。
@	从01/01/1997至今的毫秒数。

```
$（'#date'）.datepicker（{
    dateFormat： 'yy-mm-dd',
    dayNames： ['星期日', '星期一', '星期二', '星期三', '星期四', '星
    dayNamesShort： ['星期日', '星期一', '星期二', '星期三', '星期四
    dayNamesMin： ['日', '一', '二', '三', '四', '五', '六']，
    monthNames： ['一月', '二月', '三月', '四月', '五月', '六月', '
    monthNamesShort： ['一', '二', '三', '四', '五', '六', '七', '
    altField： '#abc',
    altFormat： 'yy-mm-dd',
    appendText： '（yy-mm-dd）',
    firstDay： 1，
    showWeek： true，
    weekHeader： '周',
}）；
```

表16-27 datepicker外观选项

属性	默认值/类型	说明
disabled	false/布尔值	禁用日历。
numberOfMonths	1/数值	日历中同时显示的月份个数。默认为1，如果设置3就同时显示3个月份。也可以设置数组：［3，2］，3行2列共6个。
		如果设置为true，当月中没

showOtherMonths	false/布尔值	有使用的单元格会显示填充，但无法使用。默认为false，会隐藏无法使用的单元格。
selectOtherMonths	false/布尔值	如果设置为true，表示可以选择上个月或下个月的日期。前提是show Other Months设置为true。
changeMonth	false/布尔值	如果设置为true，显示快速选择月份的下拉列表。
changeYear	false/布尔值	如果设置为true，显示快速选择年份的下拉列表。
isRTL	false/布尔值	是否由右向左绘制日历。
autoSize	false/布尔值	是否自动调整控件大小，以适应当前的日期格式的输入
showOn	'focus'/字符串	默认值为focus，获取焦点触发，还有button点击按钮触发和both任一事件发生时触发。
buttonText	'...'/字符串	触发按钮上显示的文本。
buttonImage	无/字符串	图片按钮地址。
buttonImageOnly	false/布尔值	设置为true则会使图片代替按钮。
showButtonPanel	false/布尔值	开启显示按钮面板。

closeText	'done'/字符串	设置关闭按钮的文本。
currentText	'Today'/字符串	设置获取今日日期的按钮文本。
nextText	'Next'/字符串	设置下一月的alt文本。
prevText	'Prev'/字符串	设置上一月的alt文本。
navigationAsDateFormat	false/字符串	设置prev、next和current的文字可以是format的日期格式。
yearSuffix	无/字符串	附加在年份后面的文本。
showMonthAfterYear	false/布尔值	设置为true，则将月份放置在年份之后。

```
$（'#date'）.datepicker（{
    disabled： true，
    numberOfMonths： ［3，2］，
    showOtherMonths： true，
    selectOtherMonths： true，
    changeMonth： true，
    changeYear： true，
    isRTL： true，
    autoSize： true，
    showButtonPanel： true，
    closeText： '关闭'，
    currentText： '今天'，
    showMonthAfterYear： true，
}）；
```

表16-28 datepicker日期选项

属性	默认值/类型	说明
minDate	无/对象、字符串或数值	日历中可以选择的最小日期。
maxDate	无/对象、字符串或数值	日历中可以选择的最大日期。
defaultDate	当天/日期	预设默认选定日期。没有指定，则是当天。
yearRange	无/日期	设置下拉菜单年份的区间。比如：1950：2020。
hideIfNoPrevNext	false/字符串	设置为true，如果上一月和下一月不存在，则隐藏按钮。
gotoCurrent	false/布尔值	如果为true，点击今日且回车后选择的是当前选定的日期，而不是今日。

```
$（'#date'）.datepicker（{
    yearRange：  '1950：2020',
    minDate：  -10000，
    maxDate：  0，                    //可以用new Date（2007，1，1）
    defaultDate：  -1，//可以用'1m+3'
    hideIfNoPrevNext：  true，
    gotoCurrent：  false，
}）；
```

表16-29　选择日期的字符串表示方法

属性	说明
x	当前日期之后的x天（其中x范围从1到n），比如：1，2。
-x	当前日期之前的x天（其中x范围从1到n），比

	如：-1，-2。
xm	当前日期之后的x个月（其中x范围从1到n），比如：1m，2m。
-xm	当前日期之前的x个月（其中x范围从1到n），比如：-1m，-2m。
xw	当前日期之后的x周（其中x范围从1到n），比如：1w，2w。
-xw	当前日期之后的x周（其中x范围从1到n），比如：-1w，2w。

表16-30 datepicker视觉选项

属性	默认值/类型	说明
showAnim	fadeIn/字符串	设置false，无效果。默认效果为：fadeIn。
duration	300/数值	日历显示或消失时的持续时间，单位毫秒。

```
$（'#date'）.datepicker（{
    yearRange：  '1950：2020',
    showAnim：  false，
    duration：  300，
}）；
```

表16-31 datepicker可选特效

特效名称	说明
blind	日历从顶部显示或消失。
bounce	

	日历断断续续地显示或消失，垂直运动。
clip	日历从中心垂直地显示或消失。
slide	日历从左边显示或消失。
drop	日历从左边显示或消失，有透明度变化。
fold	日历从左上角显示或消失。
highlight	日历显示或消失，伴随着透明度和背景色的变化。
puff	日历从中心开始缩放。显示时“收缩”，消失时“生长”。
scale	日历从中心开始缩放。显示时“生长”，消失时“收缩”。
pulsate	日历以闪烁形式显示或消失。
fadeIn	日历显示或消失时伴随透明度变化。

四、**datepicker**（）方法的事件

除了属性设置外，datepicker（）方法也提供了大量的事件。这些事件可以给各种不同状态时提供回调函数。这些回调函数中的this值等于对话框内容的div对象，不是整个对话框的div。

表16-32 datepicker事件选项

事件名	说明
beforeShow	日历显示之前会被调用。
	beforeShowDay（date）方法在显示日历中的

beforeShowDay	每个日期时会被调用（date参数是一个Date类对象）。该方法必须返回一个数组来指定每个日期的信息： 该日期是否可以被选择（数组的第一项，为true或false）； 该日期单元格上使用的CSS类； 该日期单元格上显示的字符串提示信息。
onChangeMonthYear	onChangeMonthYear（year，month，inst）方法在日历中显示的月份或年份改变时会被调用。或者changeMonth或changeYear为true时，下拉改变时也会触发。Year当前的年，month当年的月，inst是一个对象，可以调用一些属性获取值。
onClose	onClose（dateText，inst）方法在日历被关闭的时候调用。dateText是当时选中的日期字符串，inst是一个对象，可以调用一些属性获取值。
onSelect	onSelect（dateText，inst）方法在选择日历的日期时被调用。dateText是当时选中的日期字符串，inst是一个对象，可以调用一些属性获取值。

```
$（'#date'）.datepicker（{
    beforeShow： function （） {
        alert（'日历显示之前触发！'）；
    },
    beforeShowDay： function （date） {
        if （date.getDate（） == 1） {
            return ［false，'a'，'不能选择'］；
        } else {
            return ［true］；
        }
    },
    onChangeMonthYear： function （year，month，inst） {
        alert（year）；
    },
    onClose： function （dateText，inst） {
```

```
            alert（dateText）；
        },
        onSelect： function （dateText，inst） {
            alert（dateText）；
        }
    }）；
```

注意：JQuery UI只允许使用选项中定义的事件。目前还不可以试用on（）方法来管理。

表16-33 datepicker（'action'，param）方法

方法	返回值	说明
datepicker（'show'）	JQuery对象	显示日历。
datepicker（'hide'）	JQuery对象	隐藏日历。
datepicker（'getDate'）	JQuery对象	获取当前选定日历。
datepicker（'setDate'，date）	JQuery对象	设置当前选定日历。
datepicker（'destroy'）	JQuery对象	删除日历，直接阻断。
datepicker（'widget'）	JQuery对象	获取日历的JQuery对象。
datepicker（'isDisabled'）	JQuery对象	获取日历是否禁用。
datepicker（'refresh'）	JQuery对象	刷新一下日历。
datepicker（'option'，param）	一般值	获取options属性的值。

datepicker（'option'，param，value）	JQuery对象	设置options属性的值。

```
//显示日历
$（'#date'）.datepicker（'show'）；

//隐藏日历
$（'#date'）.datepicker（'hide'）；

//获取当前选定日期
alert（$（'#date'）.datepicker（'getDate'）.getFullYear（））；

//设置当前选定日期
$（'#date'）.datepicker（'setDate'， '2/15/2014'）；

//删除日历
$（'#date'）.datepicker（'destroy'）；

//获取日历的JQuery对象
$（'#date'）.datepicker（'widget'）；

//刷新日历
$（'#date'）.datepicker（'refresh'）；

//获取是否禁用日历
alert（$（'#date'）.datepicker（'isDisabled'））；

//获取属性的值
alert（$（'#date'）.datepicker（'option'， 'disabled'））；

//设置属性的值
$（'#date'）.datepicker（'option'， 'disabled'， true）；
```

验证插件

学习要点：

35. 使用validate.js插件

36. 默认验证规则

37. validate（）方法和选项

38. validate.js其他功能

验证插件（validate.js）是一款验证常规表单数据合法性的插件。使用它，极大地解放了在表单上繁杂的验证过程，同时也增加了用户体验。

一、使用**validate.js**插件

官网下载：http://bassistance.de/jquery-plugins/jquery-plugin-validation

最重要的文件是validate.js，还有两个可选的辅助文件：additional-methods.js（控件class方式）和message_zh.js（提示汉化）文件（实际使用，请使用min压缩版）。

第一步：引入validate.js

第二步：在JS文件中执行

$（'#reg'）.validate（）；

二、默认验证规则

Validate.js的默认验证规则的写法有两种形式：①控件属性方式；②JS键值对传参方式。

表16-34　默认规则列表

规则名	说明
required：true	必须输入字段。
email：true	必须输入正确格式的电子邮件。
url：true	必须输入正确格式的网址。
date：true	必须输入正确格式的日期（IE6验证出错）。

dateISO：true	必须输入正确格式的日期（ISO）（只验证格式，不验证有效）。
number：true	必须输入合法的数字（负数，小数）。
digits：true	必须输入正整数。
creditcard：true	必须输入合法的信用卡号，如5105105105105100。
equalTo："#field"	输入值必须和#field相同。
minlength：5	输入长度最小是5的字符串（汉字算一个字符）。
maxlength：10	输入长度最多是10的字符串（汉字算一个字符）。
rangelength：［5，10］	输入长度介于5和10之间的字符串"（汉字算一个字符）。
range：［5，10］	输入值必须介于5和10之间。
min：5	输入值不能小于5。
max：10	输入值不能大于10。
remote："check.php"	使用ajax方法调用check.php验证输入值。

```
//使用控件方式验证“必填和不得小于两位”
<input type="text" class="required" minlength="2" name="user"
```

注意：默认规则里设置布尔值的，直接写到class里即可实现。如果是数字或数组区间，直接使用属性=值的方式即可。而对于错误提示是因为可以引入中文汉化文件，或直接替换即可。

```
//使用JS对象键值对传参方式设置
```

```
$('#reg').validate({
    rules: {
        user: {                          //只有一个规则的话,
            required: true,//直接user: 'required',
            minlength: 2,
        },
    },
    messages: {
        user: {
            required: '账号不得为空!',
            minlength: '账号不得小于2位!',
        },
    }
});
```

注意：由于第一种形式不能设置指定的错误提示信息。我们推荐使用第二种形式。

```
//所有规则演示
$('#reg').validate({
    rules: {
        email: {
            email: true,
        },
        url: {
            url: true,
        },
        date: {
            date: true,
        },
        dateIOS: {
            dateIOS: true,
        },
        number: {
            number: true,
        }          digits: {
            digits: true,
        },
        creditcard: {
            creditcard: true,
        },
        min: {
            min: 5,
        },
        range: {
            range: [5, 10],
        },
```

```
        rangelength: {
            rangelength: [5, 10],
        },
        notpass: {
            equalTo: '#pass', //这里的To是大写的T
        }
    },
});
```

三、**validate**（）的方法和选项

除了默认的验证规则外，validate.js还提供了大量的其他属性和方法供开发者使用。比如调试属性，错误提示位置一系列比较有用的选项。

```
//JQuery.format格式化错误提示
$('#reg').validate({
    rules: {
        user: {
            required: true,
            minlength: 5,
            rangelength: [5, 10]
        },
    },
    messages: {
        user: {
            required: '账号不得为空!',
            minlength: JQuery.format('账号不得小于{0}位!'),
            rangelength: JQuery.format('账号限制在{0}-{1}位!'),
        },
    },
});

//开启调试模式禁止提交
$('#reg').validate({
    debug: true,
});

//设置调试模式为默认，可以禁止多个表单提交
$.validator.setDefaults({
    debug: true,
});

//使用其他方式代替默认提交
submitHandler: function (form) {
        //可以执行AJAX提交，并且无需debug:true阻止提交了
},
```

```
//忽略某个字段验证
ignore： '#pass',

//群组错误提示
groups： {
   myerror： 'user pass',
},

//显示群组的错误提示
focusInvalid： false,
errorPlacement： function （error， element） {
   $.each（error， function （index， value） {
      $（'.myerror'）.html（$（'.myerror'）.html（） + $（value
   }）
},

//群组错误提示，分开
groups： {
   error_user： 'user',
   error_pass： 'pass'
},

//将群组的错误指定存放位置
errorPlacement： function （error， element） {
   error.appendTo（'.myerror'）；
},

//设置错误提示的class名
errorClass： 'error_list',

//设置错误提示的标签
errorElement： 'p',

//统一包裹错误提示
errorLabelContainer： 'ol.error',
wrapper： 'li',

//设置成功后加载的class
success： 'success',

//使用方法加载class并添加文本
success： function （label） {
   label.addClass（'success'）.text（'ok'）；
},

//高亮显示有错误的元素，变色式
highlight： function（element， errorClass） {
```

```
        $（element）.fadeOut（function（） {
            $（element）.fadeIn（）
        }）
    },

    //高亮显示有错误的元素，变色式
    highlight： function（element， errorClass） {
        $（element）.css（'border'， '1px solid red'）；
    },

    //成功的元素移出错误高亮
    unhighlight： function （element， errorClass） {
        $（element）.css（'border'， '1px solid #ccc'）；
    },

    //表单提交时获取信息
    invalidHandler： function （event， validator） {
        var errors = validator.numberOfInvalids（）；
        if （errors） {
            $（'.myerror'）.html（'您有' + errors + '个表单元素填写非法
        }
        },

    //获取错误提示句柄，不用提交及时获取值
    showErrors： function （errorMap， errorList） {
        var errors = this.numberOfInvalids（）；
        if （errors） {
            $（'.myerror'）.html（'您有' + errors + '个表单元素填写非法
        } else {
            $（'.myerror'）.hide（）；
        }
        this.defaultShowErrors（）；            //执行默认错误
    },

    //获取错误提示句柄，errorList
    showErrors： function （errorMap， errorList） {
        alert（errorList［0］.message）；//得到错误信息
        alert（errorList［0］.element）；//当前错误的表单元素
    },
```

三、**validate.js**其他功能

使用remote：url，可以对表单进行AJAX验证，默认会提交当前验证的值到远程地址。如果需要提交其他的值，可以使用data选项。

```
    //使用AJAX验证
```

```
rules：  {
    user：  {
        required：  true，
        minlength：  2，
        remote：  'user.php'，
    }，
}，

//user.php内容
<?php
    if （$_GET［'user'］ == 'bnbbs'） {
        echo 'false'；
    } else {
        echo 'true'；
    }
?>
```

注意：远程地址只能输出'true'或'false'，不能输出其他值。

```
//同时传递多个值到远程端
pass：  {
    required：  true，
    minlength：  6，
    remote：  {
        url：  'user.php'，
        type：  'POST'，
        dataType：  'json'，
        data：  {
            user：  function （）  {
                return $（'#user'）.val（）；
            }，
        }，
    }，
}，

//user.php内容
<?php
    if （$_POST［'user'］ != 'bnbbs' || $_POST［'pass'］ != '12
    echo 'false'；
    } else {
        echo 'true'；
    }
?>
```

validate.js提供了一些事件触发的默认值，这些值大部分是不用更改的。

```
//取消提交验证
onsubmit： false，//默认是true
```

注意：设置为false会导致直接传统提交，不会实现验证功能，一般是用于keyup/click/blur验证提交。

```
//设置鼠标离开不触发验证
onfocusout： false，//默认为true

//设置键盘按下弹起不触发验证
onkeyup： false，//默认为true
```

注意：只要设置了，在测试的浏览器不管是false还是true都不触发了。

```
//设置点击checkbox和radio点击不触发验证
onclick： false，//默认为true

//设置错误提示后，无法获取焦点
focusInvalid： false，//默认为true

//提示错误时，隐藏错误提示，不能和focusInvalid一起用，冲突
focusCleanup： true，//默认为false
```

如果表单元素设置了title值，且messages为默认，就会读取title值的错误信息，我们可以通过ignoreTitle：true，设置为true，屏蔽这一个功能。

```
ignoreTitle： true，//默认为false

//判断表单所验证的元素是否全部有效
alert（$（'#reg'）.valid（））；//全部有效返回true
```

Validate.js提供了可以单独验证每个表单元素的rules方法，不但提供了add增加验证，还提供了remove删除验证的功能。

```
//给user增加一个表单验证
$（'#user'）.rules（'add'， {
    required： true，
    minlength： 2，
    messages： {
        required： '账号不得为空！'，
        minlength： JQuery.format（'账号不得小于{0}位！'），
    }
```

```
});

//删除user的所有验证规则
$('#user').rules('remove');

//删除user的指定验证规则
$('#user').rules('remove', 'minlength min max');

//添加自定义验证
$.validator.addMethod('code', function (value, element) {
    var tel = /^[0-9]{6}$/;
    return this.optional(element) || (tel.test(value));
}, '请正确填写您的邮政编码');

//调用自定义验证
rules: {
    code: {
        required: true,
        code: true,
    }
},
```

验证注册表单

学习要点：

39. html部分

40. css部分

41. JQuery部分

本节课，将使用validate.js验证插件功能，完成表单注册验证的功能。

一、**html**部分

html部分不需要更改太多，只要加个存放错误提示的列表标签即可。

<ol class="reg_error">

二、**css**部分

css部分主要是成功后引入一张小图标，还有错误列表样式。

```
#reg p .star {
    color：maroon；
}
#reg p .success {
    display：inline-block；
    width：28px；
    background：url（../img/reg_succ.png） no-repeat；
}
#reg ol {
    margin：0；
    padding：0 0 0 20px；
    color：maroon；
}
#reg ol li {
    height：20px；
}
```

三、**JQuery**部分

JQuery部分很常规，基本使用了validate.js的核心功能。

```
$（'#reg'）.dialog（{
    autoOpen： false，
    modal： true，
    resizable： false，
    width： 320，
    height： 340，
    buttons： {
        '提交'： function （） {
            $（this）.submit（）；
        }
    },
}）.buttonset（）.validate（{
    submitHandler： function （form） {
        alert（'验证完成，准备提交！'）；
    },
    showErrors： function （errorMap， errorList） {
        var errors = this.numberOfInvalids（）；
        if （errors > 0） {
            $（'#reg'）.dialog（'option'， 'height'， 20 * errors
        } else {
            $（'#reg'）.dialog（'option'， 'height'， 340）；
```

```
        }
        this.defaultShowErrors();
    },
    highlight: function (element, errorClass) {
        $(element).css('border', '1px solid #630');
    },
    unhighlight: function (element, errorClass) {
        $(element).css('border', '1px solid #ccc');
        $(element).parent().find('span').html(' ').a
    },
    errorLabelContainer: 'ol.reg_error',
    wrapper: 'li',
    rules: {
        user: {
            required: true,
            minlength: 2,
        },
        pass: {
            required: true,
            minlength: 6,
        },
        email: {
            required: true,
            email: true,
        },
        date: {
            date: true,
        },
    },
    messages: {
        user: {
            required: '账号不得为空!',
            minlength: JQuery.format('账号不得少于{0}位!'),
        },
        pass: {
            required: '密码不得为空!',
            minlength: JQuery.format('密码不得少于{0}位!'),
        },
        email: {
            required: '邮箱不得为空!',
            email: '请输入正确的邮箱格式!',
        },
        date: {
            date: '请输入正确的日期!',
        },
    },
});
```

AJAX表单插件

学习要点：

42．核心方法

43．option参数

44．工具方法

传统的表单提交需要多次跳转页面，极大地消耗了资源也缺乏良好的用户体验。而这款form.js表单的AJAX提交插件将解决这个问题。

一、核心方法

官方网站：http://malsup.com/jquery/form/

form.js插件有两个核心方法：ajaxForm（）和ajaxSubmit（）。它们集合了从控制表单元素到决定如何管理提交进行的功能。

```
//ajaxForm提交方式
$（'#reg'）.ajaxForm（function （） {
    alert（'提交成功！'）；
}）；
```

注意：使用ajaxForm（）方法，会直接实现AJAX提交。自动阻止了默认行为，而它提交的默认页面是form控件的action属性的值。提交的方式是method属性的值。

```
//ajaxSubmit（）提交方式
$（'#reg'）.submit（function （） {
    $（this）.ajaxSubmit（function （） {
        alert（'提交成功！'）；
    }）；
    return false；
}）；
```

注意：ajaxForm（）方法是针对form直接提交的，所以阻止了默认行为。而ajaxSubmit（）方法，由于是针对submit（）方法的，所以需要手动阻止默认行为。

二、**option**参数

option参数是一个以键值对传递的对象，通过这个对象，可以设置各种AJAX提交的功能。

```
$('#reg').submit(function () {
    $(this).ajaxSubmit({
        url: 'test.php',                    //设置提交的url，可覆盖acti
        target: '#box', //服务器返回的内容存放在#box里
        type: 'POST', //GET, POST
        dataType: null, //xml, json, script, 默认为null
        clearForm: true, //成功提交后，清空表单
        resetForm: true, //成功提交后，重置表单
        data: {//增加额外的数据提交
            aaa: 'bbb',
            ccc: 'ddd'.
        },
        beforeSubmit: function (formData, jqForm, options)
            alert(formData[0].name); //得到传递表单元素的name
            alert(formData[0].value); //得到传递表单元素的value
            alert(jqForm); //得到form的jquery对象
            alert(options); //得到目前options设置的属性
            alert('正在提交中!!!');
            return true;
        },
        success: function (responseText, statusText) {
            alert(responseText + statusText);    //成功后回调
        },
        error: function (event, errorText, errorType) {   /
            alert(errorText + errorType);
        },
    });
    return false;
});
```

三、工具方法

form.js除了提供两个核心方法之外，还提供了一些常用的工具方法。这些方法主要是在提交前或后对数据或表单进行处理的。

```
//表单序列化
alert($('#reg').formSerialize());

//序列化某一个字段
alert($('#reg #user').fieldSerialize());
```

```
//得到某个字段的value值
alert（$（'#reg #user'）.fieldValue（））；

//重置表单
$（'#reg'）.resetForm（）

//清空某个字段
$（'#reg #user'）.clearFields（）；
```

AJAX提交表单

学习要点：

45. 创建数据库

46. Loading制作

47. AJAX提交

本节课运用两大表单插件，完成数据表新增的工作。

一、创建数据库

创建一个数据库，名称为：zhiwen。它可以表示为：id、user、pass、email、sex、birthday、date。

所需的PHP文件：config.php、add.php、is_user.php。

```
//config.php
<?php
   header（'Content-Type：text/html； charset=utf-8'）；

   define（'DB_HOST'， 'localhost'）；
   define（'DB_USER'， 'root'）；
   define（'DB_PWD'， '123456'）；
   define（'DB_NAME'， 'zhiwen'）；

   $conn = @mysql_connect（DB_HOST， DB_USER， DB_PWD） or die

   @mysql_select_db（DB_NAME） or die（'数据库错误：'.mysql_errc

   @mysql_query（'SET NAMES UTF8'） or die（'字符集错误：'.mysql
```

```
?>

//add.php
<?php
    require 'config.php';

    $query = "INSERT INTO user （user， pass， email， sex， bi
           VALUES （'{$_POST［'user'］}'， sha1（'{$_POST［'pass

    mysql_query（$query） or die（'新增失败! '.mysql_error（））；

    echo mysql_affected_rows（）；

    mysql_close（）；
?>

//is_user.php
<?php
    require 'config.php';

    $query = mysql_query（"SELECT user FROM user WHERE user='{

    if （mysql_fetch_array（$query， MYSQL_ASSOC））{
        echo 'false';
    } else {
        echo 'true';
    }

    mysql_close（）；
?>
```

二、**Loading**的制作

在提交表单的时候，由于网络速度问题，可能会出现不同时间延迟。所以，为了更好的用户体验，在提交等待过程中，设置loading是非常有必要的。

```
//采用对话框式
$（'#loading'）.dialog（{
    modal： true，
    autoOpen： false，
    closeOnEscape： false，                    //按下esc无效
    resizable： false，
    draggable： false，
    width： 180，
    height： 50，
```

```
}）.parent（）.parent（）.find（'.ui-widget-header'）.hide（）；

//CSS部分
#loading {
    background：url（../img/loading.gif） no-repeat 20px center
    line-height：25px；
    font-size：14px；
    font-weight：bold；
    text-indent：40px；
    color：#666；
}
```

三、**AJAX**的提交

最后，我们需要采用form.js插件对数据进行提交。而且在其他部分需要做一些修改。

```
submitHandler： function （form） {
    $（form）.ajaxSubmit（{
        url： 'add.php',
        type： 'POST',
        beforeSubmit： function （formData， jqForm， options）
            $（'#loading'）.dialog（'open'）；
            $（'#reg'）.dialog（'widget'）.find（'button'）.eq（1）
        },
        success： function （responseText， statusText） {
            $（'#reg'）.dialog（'widget'）.find（'button'）.eq（1）
            if （responseText） {
                $（'#loading'）.css（'background'， 'url（img/succ
                setTimeout（function （） {
                    $（'#loading'）.dialog（'close'）；
                    $（'#loading'）.css（'background'， 'url（img/
                    $（'#reg'）.dialog（'close'）；
                    $（'#reg'）.resetForm（）；
                    $（'#reg span.star'）.html（'*'）.removeClass
                }， 1000）；
            }
        },
    }）；
```

cookie插件

学习要点：

48. 使用cookie插件

Cookie是网站用来在客户端保存识别用户的一种小文件。一般可以保存用户登录信息、购物数据信息等一系列微小信息。

一、使用**cookie**插件

```
官方网站：http：//plugins.jquery.com/cookie/
//生成一个cookie：
$.cookie（'user'， 'bnbbs'）；

//设置cookie参数
$.cookie（'user'， 'bnbbs'， {
    expires： 7，           //过期时间，7天后
    path： '/'，//设置路径，上一层
    domain： 'www.ycku.com'，//设置域名
    secure： true，//默认为false，需要使用安全协议https
}）；

//关闭编码/解码，默认为false
$.cookie.raw = true；

//读取cookie数据
alert（$.cookie（'user'））；

//读取所有cookie数据
alert（$.cookie（））；
```

注意：读取所有的cookie是以对象键值对存放的，所以，也可以从$.cookie（）.user获取。

```
//删除cookie
$.removeCookie（'user'）；

//删除指定路径cookie
$.removeCookie（'user'， {
    path： '/'，
}）；
```

二、注册直接登录

把cookie引入到知问前端中去。

```
//HTML部分
```

```
<div class="header_member">
    <a href="javascript：void（0）" id="reg_a">注册</a>
    <a href="javascript：void（0）" id="member">用户</a>
    |
    <a href="javascript：void（0）" id="login_a">登录</a>
    <a href="javascript：void（0）" id="logout">退出</a>
</div>

//JQuery部分
$（'#member， #logout'）.hide（）；

if （$.cookie（'user'））{
    $（'#member， #logout'）.show（）；
    $（'#reg_a， #login_a'）.hide（）；
} else {
    $（'#member， #logout'）.hide（）；
    $（'#reg_a， #login_a'）.show（）；
}

$（'#logout'）.click（function （）{
    $.removeCookie（'user'）；
    window.location.href = '/jquery/'；
}）；

success： function （responseText， statusText）{
    $（'#reg_a， #login_a'）.hide（）；
    $（'#member， #logout'）.show（）；
    $（'#member'）.html（$.cookie（'user'））；
},
```

选项卡**UI**

学习要点：

49．使用tabs

选项卡（tab）是一种能提供给用户在同一个页面切换不同内容的UI。尤其是在页面布局紧凑的页面上，提供了非常好的用户体验。

一、使用**tabs**

使用tabs比较简单，但需要按照指定的规范即可。

```
//HTML部分
```

```
<div id="tabs">
    <ul>
        <li><a href="#tabs1">tab1</a></li>
        <li><a href="#tabs2">tab2</a></li>
        <li><a href="#tabs3">tab3</a></li>
    </ul>
    <div id="tabs1">tab1-content</div>
    <div id="tabs2">tab2-content</div>
    <div id="tabs3">tab3-content</div>
</div>

//JQuery部分
$（'#tabs'）.tabs（）；
```

二、修改**tabs**样式

在弹出的tabs对话框中，在火狐浏览器中打开Firebug或者右击-gt;查看元素。这样，我们可以看看tabs的样式，根据样式进行修改。我们为了和网站主题符合，对tabs的标题背景进行修改。

```
//无需修改ui里的CSS，直接用style.css替代
.ui-widget-header {
    background：url（../img/ui_header_bg.png）；
}

//去掉外边框
#tabs {
    border：none；
}

//内容区域修饰
#tabs1， #tabs2， #tabs3 {
    height：100px；
    padding：10px；
    border：1px solid #aaa；
    border-top：none；
    position：relative；
    top：-2px；
}
```

三、**tabs**（）方法的属性

选项卡方法有两种形式：①tabs（options）。options是以对象键值对的形式传参，每个键值对表示一个选项。②tabs（'action'，param）。action是操作选项卡方法的字符串，param则是options的某个选项。

表16-35 tabs外观选项

属性	默认值/类型	说明
collapsible	false/布尔值	当设置为true是，允许选项卡折叠对应的内容。默认值为false，不会关闭对应内容。
disabled	无/数组	使用数组来指定禁用哪个选项卡的索引，比如［0，1］来禁用前两个选项卡。
event	click/字符串	触发tab的事件类型，默认为click。可以设置mouseover等其他鼠标事件。
active	数组和布尔值	如果是数组，初始化时默认显示哪个tab，默认值为0。如果是布尔值，那么默认是否折叠。条件必须是collapsible值为true。
heightStyle	content/字符串	默认为content，即根据内容伸展高度。Auto则自动根据最高的那个为基准，fill则是填充一定的可用高度。
show	false/布尔值	切换选项卡时，默认采用淡入效果。
hide	false布尔值	切换选项卡时，默认采用淡出效果。

```
$（'#tabs'）.tabs（{
    collapsible： true，
    disabled： ［0］，
    event： 'mouseover'，
    active： false，
```

```
    heightStyle:  'content',
    hide:  true,
    show:  true,
});
```

注意：设置true后，默认为淡入或淡出。如果想使用别的特效，可以使用以下表格中的字符串参数。

表16-36　show和hide可选特效

特效名称	说明
blind	对话框从顶部显示或消失。
bounce	对话框断断续续地显示或消失，垂直运动。
clip	对话框从中心垂直地显示或消失。
slide	对话框从左边显示或消失。
drop	对话框从左边显示或消失，有透明度变化。
fold	对话框从左上角显示或消失。
highlight	对话框显示或消失，伴随着透明度和背景色的变化。
puff	对话框从中心开始缩放。显示时“收缩”，消失时“生长”。
scale	对话框从中心开始缩放。显示时“生长”，消失时“收缩”。
pulsate	对话框以闪烁形式显示或消失。

四、**tabs**（）方法的事件

除了属性设置外，tabs（）方法也提供了大量的事件。这些事件可以给各种不同状态时提供回调函数。

表16-37 tab事件选项

事件名	说明
create	当创建一个选项卡时激活此事件。该方法有两个参数（event，ui）。ui参数有两个子属性tab和panel，得到当前活动卡和内容选项的对象。
activate	当切换一个活动卡时，启动此事件。该方法有两个参数（event，ui）。ui参数有四个子属性newTab、newPanel、oldTab，oldPanel。分别得到的时候有新tab对象、新内容对象、旧tab对象和旧内容对象。
beforeActivate	当切换一个活动卡之前，启动此事件。该方法有两个参数（event，ui）。ui参数有四个子属性newTab、newPanel、oldTab，oldPanel。分别得到的时候有新tab对象、新内容对象、旧tab对象和旧内容对象。
load	当AJAX加载一个文档后激活此事件。该方法有两个参数（event，ui）。ui参数有两个子属性tab和panel，得到当前活动卡和内容选项的对象。
beforeLoad	当ajax加载一个文档前激活此事件。该方法有两个参数（event，ui）。ui参数有四个子属性tab和panel以及jqXHR和ajaxSettings，前两个得到当前活动卡和内容选项的对象，后两个是ajax操作对象。

```
//当选项卡创建时触发
$（'#tabs'）.tabs（{
    create： function （event， ui） {
```

```
        alert（$（ui.tab.get（））.html（））；
        alert（$（ui.panel.get（））.html（））；
    },
});

//当切换到一个活动卡时触发
$（'#tabs'）.tabs（{
    activate： function （event， ui） {
        alert（$（ui.oldTab.get（））.html（））；
        alert（$（ui.oldPanel.get（））.html（））；
        alert（$（ui.newTab.get（））.html（））；
        alert（$（ui.newPanel.get（））.html（））；
    },
}）；

//当切换到一个活动卡之前触发
$（'#tabs'）.tabs（{
    beforeActivate： function （event， ui） {
        alert（$（ui.oldTab.get（））.html（））；
        alert（$（ui.oldPanel.get（））.html（））；
        alert（$（ui.newTab.get（））.html（））；
        alert（$（ui.newPanel.get（））.html（））；
    },
}）；
```

在使用load和beforeLoad事件之前，我们首先要了解一下AJAX调用的基本方法。

```
//HTML部分
<ul>
    <li><a href="tabs1.html">tab1</a></li>
    <li><a href="tabs2.html">tab2</a></li>
    <li><a href="tabs3.html">tab3</a></li>
</ul>
```

而tabs1.html、tabs2.html和tabs3.html只要书写即可，无需包含 。tabs1-content

而这个时候，我们的CSS需要做一定的修改，只要将之前的ID换成：
#ui-tabs-1， #ui-tabs-2， #ui-tabs-3 {}

```
//AJAX加载后触发
$（'#tabs'）.tabs（{
    load： function （event， ui） {
        alert（'ajax加载后触发！'）；
    }
```

```
});

//AJAX加载前触发
$('#tabs').tabs({
    beforeLoad: function (event, ui) {
        ui.ajaxSettings.url = 'tabs2.html';
        ui.jqXHR.success(function (responseText) {
            alert(responseText);
        });
    }
});
```

表16-38 tabs('action', param)方法

方法	返回值	说明
tabs('disable')	JQuery对象	禁用选项卡。
tabs('enable')	JQuery对象	启用选项卡。
tabs('load')	JQuery对象	通过AJAX获取选项卡内容。
tabs('widget')	JQuery对象	获取选项卡的JQuery对象。
tabs('destroy')	JQuery对象	删除选项卡，直接阻断了tabs。
tabs('refresh')	JQuery对象	更新选项卡，比如高度。
tabs('option', param)	一般值	获取options属性的值。
tabs('option', param, value)	JQuery对象	设置options属性的值。

```
//禁用选项卡
$('#tabs').tabs('disable');                    //$('#tabs').tab

//启用选项卡
```

```
$（'#tabs'）.tabs（'enable'）；//$（'#tabs'）.tabs（'enable'，  0

//获取选项卡JQuery对象
$（'#tabs'）.tabs（'widget'）；

//更新选项卡
$（'#tabs'）.tabs（'refresh'）；

//删除tabs选项卡
$（'#tabs'）.tabs（'destroy'）；

//重载指定选项卡的内容
$（'#button'）.click（function （） {
$（'#tabs'）.tabs（'load'，  0）；
}）；

//得到tabs的options值
alert（$（'#tabs'）.tabs（'option'，  'active'））；

//设置tabs的options值
$（'#tabs'）.tabs（'option'，  'active'，  1）；
```

五、**tabs**中使用**on**（）

在tabs的事件中，提供了使用on（）方法处理的事件方法。

表16-39 on（）方法触发的选项卡事件

特效名称	说明
tabsload	AJAX加载后触发。
tabsbeforeload	AJAX加载前触发。
tabsactivate	选项卡切换时触发。
tabsbeforeactivate	选项卡切换前触发。

```
//AJAX加载后触发
$（'#tabs'）.on（'tabsload'，  function （） {
    alert（'ajax加载后触发！'）；
}）；
```

```
//AJAX加载前触发
$（'#tabs'）.on（'tabsbeforeload'， function （） {
    alert（'ajax加载前触发！'）；
}）；

//选项卡切换时触发
$（'#tabs'）.on（'tabsactivate'， function （） {
    alert（'选项卡切换时触发！'）；
}）；

//选项卡切换前触发
$（'#tabs'）.on（'tabsbeforeactivate'， function （） {
    alert（'选项卡切换前触发！'）；
}）；
```

折叠菜单**UI**

学习要点：

50．使用accordion

折叠菜单（accordion）和选项卡一样，也是一种在同一个页面上切换不同内容的功能UI。它和选项卡的使用几乎没有什么区别，只是显示的效果有所差异罢了。

一、使用**accordion**

使用accordion比较简单，但需要按照指定的规范进行。

```
//HTML部分
＜div id="accordion"＞
    ＜h1＞菜单1＜/h1＞
    ＜div＞内容1＜/div＞
    ＜h1＞菜单2＜/h1＞
    ＜div＞内容2＜/div＞
    ＜h1＞菜单3＜/h1＞
    ＜div＞内容3＜/div＞
＜/div＞

//JQuery部分
$（'#accordion'）.accordion（）；
```

二、修改**accordion**样式

在显示的accordion折叠菜单中，在火狐浏览器中打开Firebug或者右击->查看元素。这样，我们可以看看accordion的样式，根据样式进行修改。我们为了和网站主题符合，对accordion的标题背景进行修改。

```
//无需修改ui里的CSS，直接用style.css替代
.ui-widget-header {
   background：url（../img/ui_header_bg.png）；
}
```

三、**accordion**（）方法的属性

选项卡方法有两种形式：①accordion（options）。options是以对象键值对的形式传参，每个键值对表示一个选项。②accordion（'action'，param）。action是操作选项卡方法的字符串，param则是options的某个选项。

表16-40　accordion外观选项

属性	默认值/类型	说明
collapsible	false/布尔值	当设置为true时，允许菜单折叠对应的内容。默认值为false，不会关闭对应内容。
disabled	无/布尔值	默认为false，设置为true则禁用折叠菜单。
event	click/字符串	触发accordion的事件类型，默认为click。可以设置mouseover等其他鼠标事件。
active	数组和布尔值	如果是数组，初始化时默认显示哪个tab，默认值为0。如果是布尔值，那么默认是否折叠。条件必须是collapsible值为true。

heightStyle	content/字符串	默认为auto，即自动根据最高的那个为基准，fill则是填充一定的可用高度，content则是根据内容伸展高度。
header	h1/字符串	设置折叠菜单的标题标签。
icons	默认图标	设置想要的图标。

```
$（'#accordion'）.accordion（{
    collapsible： true，
    disabled： true，
    event： 'mouseover'，
    active： 1，
    active： true，
    heightStyle： 'content'，
    header： 'h3'，
    icons： {
        "header"： "ui-icon-plus"，
        "activeHeader"： "ui-icon-minus"，
    }，
}）；
```

四、**accordion**（）方法的事件

除了属性设置外，accordion（）方法也提供了大量的事件。这些事件可以给各种不同状态时提供回调函数。

表16-41 accordion事件选项

事件名	说明
create	当创建一个折叠菜单时激活此事件。该方法有两个参数（event，ui）。ui参数有两个子属性header和panel，得到当前标题和内容选项的对象。
	当切换一个折叠菜单时，启动此事件。该方法有两个参数（event，ui）。ui参数有四个子属

activate	性newHeader、newPanel、oldHeader，oldPanel。分别得到的时候有新header对象、新内容对象、旧header对象和旧内容对象。
beforeActivate	当切换一个折叠菜单之前，启动此事件。该方法有两个参数（event，ui）。ui参数有四个子属性newHeader、newPanel、oldHeader，oldPanel。分别得到的时候有新header对象、新内容对象、旧header对象和旧内容对象。

```
//当折叠菜单创建时触发
$('#accordion').accordion({
    create: function (event, ui) {
        alert($(ui.header.get()).html());
        alert($(ui.panel.get()).html());
    },
});

//当切换到一个菜单时触发
$('#accordion').accordion({
    activate: function (event, ui) {
        alert($(ui.oldHeader.get()).html());
        alert($(ui.oldPanel.get()).html());
        alert($(ui.newHeader.get()).html());
        alert($(ui.newPanel.get()).html());
    },
});

//当切换到一个菜单之前触发
$('#accordion').accordion({
    beforeActivate: function (event, ui) {
        alert($(ui.oldHeader.get()).html());
        alert($(ui.oldPanel.get()).html());
        alert($(ui.newHeader.get()).html());
        alert($(ui.newPanel.get()).html());
    },
});
```

表16-42 accordion('action', param)方法

方法	返回值	说明

accordion（'disable'）	JQuery对象	禁用折叠菜单。
accordion（'enable'）	JQuery对象	启用折叠菜单。
accordion（'widget'）	JQuery对象	获取折叠菜单的JQuery对象。
accordion（'destroy'）	JQuery对象	删除折叠菜单，直接阻断了accordion。
accordion（'refresh'）	JQuery对象	更新折叠菜单，比如高度。
accordion（'option'，param）	一般值	获取options属性的值。
accordion（'option'，param，value）	JQuery对象	设置options属性的值。

```
//禁用折叠菜单
$（'#accordion'）.accordion（'disable'）；

//启用折叠菜单
$（'#accordion'）.accordion（'enable'）；

//获取折叠菜单JQuery对象
$（'#accordion'）.accordion（'widget'）；

//更新折叠菜单
$（'#accordion'）.accordion（'refresh'）；

//删除accordion折叠菜单
$（'#accordion'）.accordion（'destroy'）；

//得到accordion的options值
alert（$（'#accordion'）.accordion（'option'， 'active'））；

//设置accordion的options值
$（'#accordion'）.accordion（'option'， 'active'， 1）；
```

五、**accordion**中使用**on**（）

在accordion的事件中，提供了使用on（）方法处理的事件方法。

表16-43　on（）方法触发的选项卡事件

特效名称	说明
accordionactivate	折叠菜单切换时触发。
accordionbeforeactivate	折叠菜单切换前触发。

```
//菜单切换时触发
$（'#accordion'）.on（'accordionactivate', function （） {
    alert（'菜单切换时触发！'）；
}）；

//菜单切换前触发
$（'#accordion'）.on（'accordionbeforeactivate', function （）
    alert（'菜单切换前触发！'）；
}）；
```

编辑器插件

学习要点：

51．编辑器简介

编辑器（Editor）一般用于类似于word一样的文本编辑器，只不过是编辑为HTML格式的。分类纯JS类型的，还有JQuery插件类型的。

一、编辑器简介

我们使用的JQuery版本比较新，但尚未全面使用2.0的版本，因为IE6、IE7、IE8被抛弃了。而恰恰在这个时期，就出现编辑器插件的两极分化的局面。高端和先进的HTML编辑器已经全面不支持IE6、IE7、IE8了，而老版本的HTML编辑器在使用JQuery1.10.x版本时会发生这样或那样的不兼容。为此，还需要引入jquery-migrate-1.2.1.js向下兼容插件才能解决一部分问题。并且需要手动修改源代码才能保证其正常运行。

二、引入**uEditor**

第一步：引入jquery-migrate-1.2.1.js文件，排除编辑器低版本的问题。

第二步：把编辑器文件夹包放入根目录下。

第三步：引入uEditor.js和uEditor.css两个文件。

第四步：插入textarea，设置规定值。

第五步：JQuery调用运行。

```
//HTML部分代码
<button id="question_button">提问</button>

<form id="question" action="123.html" method="post" title="提
    <p>
        <label for="user">问题名称：</label>
        <input type="text" name="title" class="text" style="wi
        <span class="star"></span>

    <p>
        <textarea class="uEditorCustom" name="content">请填写问

</form>

<div id="error">请登录后操作...</div>

//JQuery部分代码
    $('#question').dialog({
        autoOpen: false,
        modal: true,
        resizable: false,
        width: 500,
        height: 360,
        buttons: {
            '发布': function () {
                $(this).submit();
            }
        }
    });

    $('.uEditorCustom').uEditor();

    $('#error').dialog({
        autoOpen: false,
```

```
        modal： true，
        closeOnEscape： false，
        resizable： false，
        draggable： false，
        width： 180，
        height： 50，
    }）.parent（）.find（'.ui-widget-header'）.hide（）；
```

AJAX提问

学习要点：

52. AJAX提问

本节课主要是创建一个问题表，将提问数据通过Ajax方式提交出去；然后对内容显示进行布局，实现内容部分隐藏和完整显示的功能。

首先，创建一个数据表：question，分别建立：id、title、content、user、date。

然后，创建一个PHP文件：add_content.php

```
//新增提问代码
<?php
    sleep（3）；
    require 'config.php'；
    $query = "INSERT INTO question （title， content， user， d
        VALUES （'{$_POST［'title'］}'， '{$_POST［'content'］}'，
    mysql_query（$query） or die（'新增失败! '.mysql_error（））；
    echo mysql_affected_rows（）；
    mysql_close（）；
?>

//JQuery代码
$（'#question'）.dialog（{
    autoOpen： false，
    modal： true，
    resizable： false，
    width： 500，
    height： 360，
    buttons： {
        '发布'： function （） {
            $（this）.ajaxSubmit（{
                url： 'add_content.php'，
```

```
                data: {
                    user: $.cookie('user'),
                    content: $('.uEditorIframe').contents().f
                },
                beforeSubmit: function (formData, jqForm, opt
                    $('#loading').dialog('open');
                    $('#question').dialog('widget').find('but
                },
                success: function (responseText, statusText)
                    if (responseText) {
$('#question').dialog('widget').find('button').eq(1).but
                        $('#loading').css('background', 'url(i
                        setTimeout(function () {
                            $('#loading').dialog('close');
                            $('#question').dialog('close');
                            $('#question').resetForm();
                            $('.uEditorIframe').contents().find
                            $('#loading').css('background', 'ur
                        }, 1000);
                    }
                },
            });
        }
    }
});
```

AJAX显示

学习要点：

53. AJAX显示

54. 使用字符串截取

本节课需要从数据库里获取相应数据，然后转换为JSON模式，最终在页面上显示出来。我们希望显示的时候隐藏大部分，显示摘要，具有切换功能。

一、AJAX显示

```
//从服务器端获取数据，转化为JSON格式
<?php
   require 'config.php';
```

```
    $query = mysql_query（"SELECT title，content，user，date FR

    $json = ''；

    while （!!$row = mysql_fetch_array（$query， MYSQL_ASSOC））
        foreach （ $row as $key => $value ） {
            $row［$key］ = urlencode（str_replace（"\\n"，""， $v
        }
        $json .= urldecode（json_encode（$row））.', '；
    }

    echo '［'.substr（$json， 0， strlen（$json） - 1）.'］'；

    mysql_close（）；

?＞

//JQuery部分
$.ajax（{
    url： 'show_content.php',
    type： 'POST',
    success： function （response， status， xhr） {
        var json = $.parseJSON（response）；
        var html = ''；
        var arr = ［］；
        for （var i = 0； i ＜ json.length； i ++） {
            html += '＜h4＞' + json［i］.user + ' 发表于 ' + json
        }
        $（'.content'）.append（html）；

        $.each（$（'.editor'）， function （index， value） {
            arr［index］ = $（value）.height（）；
            if （$（value）.height（） ＞ 200） {
                $（value）.next（'.bottom'）.find（'.up'）.hide（）
            }
            $（value）.height（155）；
        }）；

        $.each（$（'.bottom .down'）， function （index， value）
            $（this）.click（function （） {
                $（this）.parent（）.prev（）.height（arr［index］）
                $（this）.hide（）；
$（this）.parent（）.find（'.up'）.show（）；
            }）；
        }）；

        $.each（$（'.bottom .up'）， function （index， value） {
```

```
            $（this）.click（function （） {
                $（this）.parent（）.prev（）.height（155）；
                $（this）.hide（）；
                $（this）.parent（）.find（'.down'）.show（）；
            }）；
        }）；
    },
}）；

//CSS部分
.content h4 {
    color：#666；
    font-weight：normal；
}
.content h3 {
    color：#369；
}
.content .editor {
    color：#333；
    line-height：180%；
    /*height：110px；*/
    overflow：hidden；
}
.content .bottom {
    padding：5px 0 0 0；
}
.content hr {
    color：#ccc；
    height：1px；
}
.content .up {
    float：right；
    color：#369；
    cursor：pointer；
}
.content .down {
    float：right；
    color：#369；
    cursor：pointer；
}
```

二、使用字符串截取

由于第一种方法可能会造成溢出的字体成半截形式，所以，我们将使用另外一一种方法，即字符串截取的方式来实现这个功能。

```
//JQuery代码
var json = $.parseJSON（response）；
var html = ''；
var arr = []；
var summary = []；
for （var i = 0； i ＜ json.length； i ++） {
    html += '＜h4＞' + json［i］.user + ' 发表于 ' + json［i］.da
}
$（'.content'）.append（html）；

$.each（$（'.editor'）， function （index， value） {
    arr［index］ = $（value）.html（）；
    summary［index］ = arr［index］.substr（0，200）；
    if （summary［index］.substring（199，200） == '＜'） {
        summary［index］ = replacePos（summary［index］， 200， '
    }
    if （summary［index］.substring（198，200） == '＜/'） {
        summary［index］ = replacePos（summary［index］， 200， '
        summary［index］ = replacePos（summary［index］， 199， '
    }
    if （arr［index］.length ＞ 200） {
        summary［index］ += '...＜span class="down"＞显示全部＜/sp
        $（value）.html（summary［index］）；
    }
    $（value）.next（'.bottom'）.find（'.up'）.hide（）；
}）；

$.each（$（'.editor'）， function （index， value） {
    $（this）.on（'click'， '.down'， function （） {
        $（'.bottom .up'）.eq（index）.show（）；
        $（'.editor'）.eq（index）.html（arr［index］）；
        $（this）.hide（）；
    }）；
}）；

$.each（$（'.bottom'）， function （index， value） {
    $（this）.on（'click'， '.up'， function （） {
        $（'.editor'）.eq（index）.html（summary［index］）；
        $（this）.hide（）；
    }）；
}）；

//替换特殊字符的函数
function replacePos（strObj， pos， replacetext） {
    var str = strObj.substr（0， pos-1） + replacetext + strObj
    return str；
}
```

AJAX提交评论

学习要点：

54. 显示评论表单

55. 提交评论

本节课主要实现AJAX评论功能，包括AJAX显示评论、提交评论、加载更多等操作。

一、显示评论表单

```
//点击评论按钮展开表单
$.each（$（'.bottom'），function （index，value） {
    $（this）.on（'click'，'.comment'，function （） {
        if （$.cookie（'user'）） {
            if （!$（'.comment_list'）.eq（index）.has（'form'）.
                $（'.comment_list'）.eq（index）.append（'＜form＞
class="comment_add"＞＜dt＞＜textarea type="text"
name="comment"＞＜/textarea＞＜/dt＞＜dd＞＜input type="hidden"
$（this）.attr（'data-id'） + '" /＞＜input type="hidden" name=
'" /＞' + $.cookie（'user'） + '＜input type="button" value="发
            }
            if （$（'.comment_list'）.eq（index）.is（':hidden'）
                $（'.comment_list'）.eq（index）.show（）；
            } else {
                $（'.comment_list'）.eq（index）.hide（）；
            }
        } else {
            $（'#error'）.dialog（'open'）；
            setTimeout（function （） {
                $（'#error'）.dialog（'close'）；
                $（'#login'）.dialog（'open'）；
            }，1000）；
        }
    }）；
}）；

//评论按钮增加两个内容
＜span class="comment" data-id="' + json［i］.id + '" data-use
0条评论＜/span＞
```

二、提交评论

创建数据表：comment，字段为id、titleid、user、comment、date等。

```
//add_comment.php
<?php
    sleep（1）；
    require 'config.php'；

    $query = "INSERT INTO comment （titleid， comment， user，
           VALUES （'{$_POST［'titleid'］}'， '{$_POST［'comment

    mysql_query（$query） or die（'新增失败! '.mysql_error（））；

    echo mysql_affected_rows（）；

    mysql_close（）；
?>

//AJAX提交评论
$（'.comment_list'）.eq（index）.find（'input［type=button］'）.
    var _this = this；
    $（'.comment_list'）.eq（index）.find（'form'）.ajaxSubmit（
        url： 'add_comment.php'，
        type： 'POST'，
        beforeSubmit： function （formData， jqForm， options）
            $（'#loading'）.dialog（'open'）；
            $（_this）.button（'disable'）；
        }，
        success： function （responseText， statusText） {
            if （responseText） {
                $（_this）.button（'enable'）；
                $（'#loading'）.css（'background'， 'url（img/succ
                setTimeout（function （） {
                       $（'#loading'）.dialog（'close'）；
                       $（'#loading'）.css（'background'， 'url（i
                       $（'.comment_list'）.eq（index）.find（'for
                   }， 1 000）；
        }
    }，
}）；

//CSS部分
.content .comment {
    color：#369；
    cursor：pointer；
}
.content .comment_list {
```

```
    display：none；
    border-radius：4px；
    border：1px solid #ccc；
    min-height：25px；
    padding：5px 10px；
}
.content .comment_list dl {
    margin：0；
    padding：3px 10px 5px 0；

}
.content .comment_list dl dt {
    margin：0；
    padding：5px 0 0 0；
    color：#369；
}
.content .comment_list dl dd {
    margin：0；
    padding：0；
    line-height：180%；
    color：#333；
}
.content .comment_list dl dd.date {
    color：#999；
}
.content .comment_add {
    border：none；
    text-align：right；
}
.content .comment_add textarea {
    width：100%；
    border：1px solid #ccc；
    background：#fff；
    padding：5px；
    margin：0 0 5px 0；
    border-radius：4px；
    font-size：12px；
    color：#666；
    resize：none；
}
.content .comment_add input {
    cursor：pointer；
    position：relative；
    right：-5px；
}
.content .comment_content {
    border-bottom：1px solid #ccc；
}
```

```
.content .comment_load dd {
    background：url（../img/comment_load.gif） no-repeat 75px 5
}
```

AJAX显示评论

学习要点：

56. 显示评论

57. 提交评论

本节课主要实现AJAX评论功能，包括AJAX显示评论、提交评论、加载更多等操作。

```
//显示共显示了多少条评论
$query = mysql_query（"SELECT （SELECT COUNT（*） FROM comment

//显示评论的部分JQuery代码
var comment_this = this；
if （!$（'.comment_list'）.eq（index）.has（'form'）.length） {
    $.ajax（{
        url： 'show_comment.php',
        type： 'POST',
        data： {
            titleid： $（comment_this）.attr（'data-id'），
        },
        beforeSend： function （jqXHR， settings） {
            $（'.comment_list'）.eq（index）.append（'＜dl class=
        },
        success： function （response， status） {
            $（'.comment_list'）.eq（index）.find（'.comment_loac
            var json_comment = $.parseJSON（response）；
            $.each（json_comment， function （index2， value） {
                $（'.comment_list'）.eq（index）.append（'＜dl
class="comment_content"＞＜dt＞' + value.user + '＜/dt＞＜dd＞'
value.date + '＜/dd＞＜/dl＞'）；
            }）；
        }）；
    }）；
}

//提交评论，自动显示
var date = new Date（）；
```

```
$（'.comment_list'）.eq（index）.prepend（'＜dl class="comment_

//loading样式
.content .comment_load dd {
   background：url（../img/comment_load.gif） no-repeat 75px 4
}
```

AJAX加载更多

学习要点：

58．服务器端分页

59．客户端加载

本节课主要实现AJAX评论功能，包括AJAX显示评论、提交评论、加载重点更多等操作。

一、服务器端分页

```
//show_comment.php
＜?php
   sleep（1）；
   require 'config.php'；


   $_sql = mysql_query（"SELECT COUNT（*） AS count FROM comme
   $_result = mysql_fetch_array（$_sql， MYSQL_ASSOC）；

   $_page = 1；
   $_pagesize = 2；
   $_count = ceil（$_result［'count'］ / $_pagesize）；

   if （!isset（$_POST［'page'］））{
      $_page = 1；
   } else {
      $_page = $_POST［'page'］；
      if （$_page ＞ $_count） {
         $_page = $_count；
      }
   }

   $_limit = （$_page - 1） * $_pagesize；
```

```
    $query = mysql_query（"SELECT （{$_count}） AS count，title
    WHERE titleid='{$_POST［'titleid'］}' ORDER BY date DESC LI
```

二、客户端加载

```
//JQuery加载更多代码
success： function （response， status） {
    $（'.comment_list'）.eq（index）.find（'.comment_load'）.hid
    var count = 0；
    var json_comment = $.parseJSON（response）；
    $.each（json_comment， function （index2， value） {
        count = value.count；
    $（'.comment_list'）.eq（index）.append（'＜dl class="commen
    }）；
    $（'.comment_list'）.eq（index）.append（'＜dl＞＜dd＞＜span
    var page = 2；
    if （page ＞ count） {
        $（'.comment_list'）.eq（index）.find（'.load_more'）.of
        $（'.comment_list'）.eq（index）.find（'.load_more'）.hi
    }
    $（'.comment_list'）.eq（index）.find（'.load_more'）.button
    $（'.comment_list'）.eq（index）.find（'.load_more'）.button
        $.ajax（{
            url： 'show_comment.php'，
            type： 'POST'，
            data： {
                titleid： $（comment_this）.attr（'data-id'），
                page： page，
            }，
            beforeSend： function （jqXHR， settings） {
                $（'.load_more'）.html（'＜img src="img/more_load
            }，
            success： function （response， status） {
                var json_comment_more = $.parseJSON（response）；
                $.each（json_comment_more， function （index3， v
                    $（'.comment_list'）.eq（index）.find（'.comme
                dl class="comment_content"＞＜dt＞' + value.user
                value.comment + '＜/dd＞＜dd class="date"＞' + va
            }）；
            $（'.load_more'）.html（'加载更多评论'）；
            $（'.comment_list'）.eq（index）.find（'.load_more'）
        page++；
        if （page ＞ count） {
            $（'.comment_list'）.eq（index）.find（'.load_more'）
            $（'.comment_list'）.eq（index）.find（'.load_more'）
        }
    }，
```

```
  });
});

//CSS部分
.content .load_more {
    width: 100%;
    margin: 10px 0 0 0;
    height: 30px;
    line-height: 30px;
}
.content .load_more img {
    padding: 5px 0 0 0;
}
```

总结及屏蔽低版**IE**

学习要点：

60. 搜索功能

61. 切换功能

62. footer

63. 屏蔽低版本IE

本节课主要总结一下尚未完成的功能，以及屏蔽掉IE6、IE7等低版浏览器的支持。

一、搜索功能

搜索功能，可以使用自动补全UI+按键弹起事件+AJAX查询即可。

二、切换功能

这里可以直接设置最热门、推荐、评论最多的提问。

三、**Footer**

填充一下footer。

四、屏蔽低版本**IE**

＜!--［if lt IE 8］＞＜script＞
window.location.href="/jquery/error/"＜/script＞＜!［endif］--＞

五、总结

本书讲解了JQuery的基础和JQuery UI的基础用法。更多的实战小案例和高级技巧，我们将在扩展课程中为大家展示。

参考文献

[1] 陶国荣. JQuery权威指南 [M] . 北京：机械工业出版社，2011.

[2] [美] Bear Bibeault Yehuda Katz. JQuery实战 [M] . 2版. 北京：人民邮电出版社，2012.

[3] 单东林，等. 锋利的JQuery [M] . 2版. 北京：人民邮电出版社，2012.