征服 Ajax

Dojo、Prototype、script.aculo.us 框架解析与实例

施伟伟　张　蓓　编著

附源代码光盘
CD-ROM

人民邮电出版社
POSTS & TELECOM PRESS

征服 Ajax

Dojo、Prototype、script.aculo.us 框架解析与实例

施伟伟　张　蓓　编著

人民邮电出版社
北京

图书在版编目（CIP）数据

征服 Ajax：Dojo、Prototype、script.aculo.us 框架解析与实例 / 施伟伟，张蓓编著.
—北京：人民邮电出版社，2007.3

ISBN 978-7-115-15803-1

Ⅰ. 征… Ⅱ. ①施… ②张… Ⅲ. 计算机网络—程序设计 Ⅳ. TP393.09

中国版本图书馆 CIP 数据核字（2007）第 014185 号

内 容 提 要

运用已有的成熟框架进行项目开发，不但能提高代码的稳定性和兼容性，更能大大缩短项目的开发时间。本书详细讲解 3 个著名的 Ajax 开发框架——Dojo、Prototype 和 script.aculo.us。在介绍框架的原理和使用方法的基础上，提供了典型的应用案例。

全书分为 8 章，第 1 章介绍 Ajax 技术的基本概念及技术基础；第 2、3 章分别介绍 Prototype 框架和 script.aculo.us 框架；第 4、5 章配合使用 Prototype 和 script.aculo.us 框架实现了两个 Ajax 应用实例——网络书签和个性化主页；第 6 章对 Dojo 开发工具包的原理和使用方法进行了详细介绍；第 7 章介绍关于 Dojo 的高级话题；第 8 章使用 Dojo 开发工具包实现了一个具有基本功能的电子邮件系统。通过对本书的学习，读者不但能了解和掌握 Dojo、Prototype 和 script.aculo.us 开发框架，而且能以书中所提供的实例为原型，快速运用成熟框架开发类似应用系统。

本书结构清晰，实用性强，适合作为 Web 开发人员的参考用书。

征服 Ajax——Dojo、Prototype、script.aculo.us 框架解析与实例

◆ 编　　著　施伟伟　张　蓓
责任编辑　李　岚

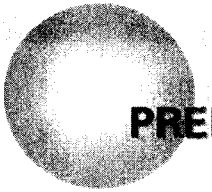
◆ 人民邮电出版社出版发行　　北京市崇文区夕照寺街 14 号
邮编　100061　　电子函件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
北京顺义振华印刷厂印刷
新华书店总店北京发行所经销

◆ 开本：800×1000　1/16
印张：22.5
字数：493 千字　　2007 年 3 月第 1 版
印数：1－5 000 册　　2007 年 3 月北京第 1 次印刷

ISBN 978-7-115-15803-1/TP

定价：49.00 元（附光盘）

读者服务热线：(010)67132692　印装质量热线：(010)67129223



前　言

在当今的 Web 应用开发中，Ajax 无疑是一个让人眼前一亮的名词。有人说 Ajax 是“新瓶装旧酒”，这句话有它一定的道理：基于 Ajax 的 Web 应用中使用的技术，如 XMLHttpRequest、JavaScript、CSS 等，确实是 Web 开发领域中的“旧”技术。而 Ajax 的“新”，体现在它综合运用这些“旧”技术的方式，以及这种运用方式所带来的独特用户体验。Ajax 的成功之处就在于此。

但是，并不是在 Web 应用中使用 Ajax 就一定会带来用户体验的提升，任何一种技术或者技巧的使用都需要因地制宜。在使用 Ajax 的方式上，已经有先行者为我们总结了一些行之有效的应用模式，比如拖曳、局部更新等。这些应用模式可能会在一个 Web 应用程序中反复地出现，这就意味着我们可能需要在页面上重复编写相同或者类似的代码。如何实现这些代码的“复用”，而不是简单的“复制”，这是 Ajax 应用开发中亟待解决的问题。

在开发 Ajax 应用中需要解决的另一个问题是代码的稳定性和兼容性。编写兼容各种浏览器的 JavaScript 代码是很困难的，不同的浏览器使用了不同的 JavaScript 解释器，它们对于同一段 JavaScript 代码的表现行为可能会不相同。尽管这类不兼容现象出现的可能性并不大，但是只需出现一次，便足以导致整个应用程序不能正常工作。

为了解决上面提到的两个问题，各种 Ajax 开发框架应运而生，它们对各种常用功能进行封装，屏蔽浏览器之间的差别。本书将要详细讲解其中的 3 个著名框架：Prototype、script.aculo.us 和 Dojo。Prototype 和 script.aculo.us 通常配合一同使用，它们的特点是短小精悍，适合在中小型项目使用；Dojo 则是一个功能十分强大的 Ajax 开发工具包，适合在企业级的项目中应用。

本书的第 1 章介绍了 Ajax 技术的基本概念及其必需的基础技术，如果您不了解 Ajax，不知道 XMLHttpRequest 对象的使用方法，可以通过阅读第 1 章对 Ajax 技术进行初步的了解。

第 2 章对 Prototype 框架进行了深入的分析和讲解，包括 Prototype 的常用函数、Ajax 相关功能，以及 Prototype 相关对象和类的详细参考。

第 3 章介绍的框架是 script.aculo.us，它提供了非常丰富的页面特效，本书对其特效的实现原理、调用方法进行了详细讲解。

第 4 章和第 5 章分别是基于 Prototype 和 script.aculo.us 框架实现的 Ajax 应用实例——网络书签和个性化主页，读者可以通过这两个实例进一步深化第 2 章和第 3 章学习的内容。同时这两章的实例已经具有一定的实用性，可以作为开发类似应用系统的原型。

第 6 章对 Dojo 开发工具包的原理和使用方法进行了详细介绍，在相关功能的介绍过程中均配有短小的示例，读者可以边学边练。

第 7 章介绍了 Dojo 相关的高级话题，如 Dojo 代码打包系统、压缩工具的使用，扩展自定义 Widget 的方法等，第 7 章是对第 6 章内容的补充和深化。

第 8 章使用 Dojo 开发工具包实现了一个具有基本功能的电子邮件系统，其中综合运用了 Dojo 的各类组件，是非常不错的 Dojo 应用范例。

最后，本书的附录对 Ajax 相关的开发工具进行了简要介绍，供有需要的读者参考。

本书主要由施伟伟、张蓓编写，由于时间仓促和作者的水平有限，书中错误和不妥之处在所难免，敬请读者批评指正。如果有任何问题，可以发送 E-mail 到 shiweiwei97@gmail.com，作者将尽快予以答复。

编　者

2007 年 2 月

CONTENTS

目　录

第 1 章　Ajax 基础……1
1.1　Ajax 简介……1
1.2　Ajax 开发需注意的几个问题……3
1.2.1　浏览器兼容性问题……3
1.2.2　XMLHttpRequest 对象封装……4
1.2.3　简化客户端脚本开发……7
1.2.4　中文编码问题……8
1.3　Ajax 开发关键技术……8
1.3.1　XMLHttpRequest 对象……9
1.3.2　JavaScript 面向对象特性……15
1.3.3　JavaScript 面向对象编程实现……20
1.3.4　文档对象模型（DOM）……25
1.4　主流 Ajax 开发框架介绍……28
1.4.1　浏览器端框架……28
1.4.2　基于服务器端的框架……30
1.5　小结……32

第 2 章　Prototype 框架……33
2.1　Prototype 框架简介……33
2.1.1　什么是 Prototype……33
2.1.2　Prototype 的获取和使用……34
2.2　常用函数……34
2.2.1　$()函数……34
2.2.2　$A()函数……35
2.2.3　$F()函数……37
2.2.4　$H()函数……38

2.2.5 $R()函数……38
2.2.6 $$()函数……40
2.2.7 Try.these()函数……42
2.3 Prototype 对 Ajax 的支持……43
2.3.1 Ajax 对象……44
2.3.2 Ajax.Base 类……44
2.3.3 Ajax.Request 类……44
2.3.4 Ajax.Updater 类……46
2.3.5 Ajax.PeriodicalUpdater 类……47
2.3.6 Ajax.Responders 对象……50
2.4 Prototype 对象参考……51
2.4.1 基础类……52
2.4.2 字符串处理（String 对象扩展）……54
2.4.3 枚举对象（Enumerable 对象）……57
2.4.4 数组（Array 对象扩展）……64
2.4.5 Hash 对象……65
2.4.6 对象范围类（ObjectRange 类）……67
2.4.7 DOM 扩展……67
2.4.8 CSS 选择符（Selector 类）……72
2.4.9 表单支持……72
2.4.10 事件处理（Event 对象扩展）……77
2.4.11 位置处理（Position 对象）……80
2.5 小结……81

第 3 章 script.aculo.us 框架……82
3.1 script.aculo.us 简介……82
3.2 script.aculo.us 基础工具类……83
3.3 script.aculo.us 特效……84
3.3.1 基本特效……84
3.3.2 组合特效……89
3.4 script.aculo.us 高级功能……98
3.4.1 拖放效果……98
3.4.2 滑块控件……107
3.4.3 Ajax 控件集……110
3.5 小结……117

第 4 章 Prototype 和 script.aculo.us 应用实例——网络书签 ····· 119
4.1 需求分析 ····· 119
4.1.1 什么是网络书签 ····· 119
4.1.2 功能需求 ····· 121
4.2 数据库设计 ····· 121
4.2.1 数据表设计 ····· 121
4.2.2 存储过程设计 ····· 122
4.3 系统设计与实现 ····· 130
4.3.1 显示书签分类列表 ····· 131
4.3.2 移动书签分类 ····· 132
4.3.3 显示书签列表 ····· 135
4.3.4 移动书签 ····· 141
4.3.5 删除书签 ····· 142
4.3.6 添加书签分类 ····· 144
4.3.7 删除书签分类 ····· 146
4.3.8 修改书签分类名称 ····· 147
4.3.9 添加书签 ····· 148
4.3.10 修改书签 ····· 151
4.4 小结 ····· 153

第 5 章 Prototype 和 script.aculo.us 应用实例——个性化主页 ····· 155
5.1 应用背景 ····· 155
5.1.1 个性化主页 ····· 155
5.1.2 RSS 介绍 ····· 157
5.2 需求分析 ····· 158
5.3 系统设计与实现 ····· 159
5.3.1 页面布局 ····· 159
5.3.2 拖曳效果实现 ····· 162
5.3.3 个性化主页栏目的呈现 ····· 164
5.3.4 记忆栏目定制信息 ····· 171
5.3.5 页面更新提示 ····· 176
5.3.6 栏目定时更新 ····· 177
5.4 小结 ····· 180

第 6 章 Dojo 开发工具包 ····· 181
6.1 Dojo 简介 ····· 182

6.1.1 Dojo 的历史和发展……182
6.1.2 获取 Dojo 版本……182
6.1.3 动态加载 package……184
6.1.4 Dojo 的架构……185
6.1.5 Dojo 相关资源……186
6.2 Dojo 基础工具类……186
6.2.1 类的声明和继承……187
6.2.2 数组迭代器……190
6.2.3 数据加密/解密……191
6.2.4 数学函数……191
6.2.5 字符串处理……194
6.2.6 集合类数据结构……197
6.2.7 日期、时间处理函数……199
6.3 Dojo 的事件系统……202
6.3.1 基于 Dojo 的事件绑定……202
6.3.2 基于 Dojo 的面向切面编程……204
6.3.3 事件的发布和订阅……207
6.3.4 页面 onload 和 onunload 事件……207
6.4 Dojo 的用户界面组件……208
6.4.1 DOM 操作……208
6.4.2 拖曳效果支持……210
6.4.3 页面特效……219
6.5 Dojo 的 Ajax 组件……225
6.5.1 页面异步调用……226
6.5.2 文件上传……227
6.5.3 跨域页面调用……228
6.5.4 远程过程调用……231
6.6 Dojo 的表单处理组件……234
6.6.1 表单绑定……234
6.6.2 表单验证……235
6.7 Dojo 的存储组件……238
6.8 Dojo 的图表绘制组件……239
6.9 Dojo Widget……243
6.9.1 Layout Widget……244
6.9.2 Form Widget……249

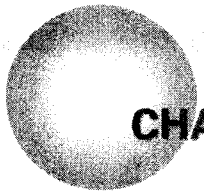
6.9.3 General Widget ……253
6.10 小结……265

第 7 章 深入 Dojo 开发……267
7.1 Dojo 的代码打包系统……268
7.1.1 下载完整的 Dojo 开发包……268
7.1.2 安装 JDK 和 Ant……269
7.1.3 定制 Dojo 代码发布包……270
7.1.4 Dojo 代码打包系统参数说明……272
7.2 Dojo 的代码压缩工具……273
7.3 Dojo 的 JavaScript 链接工具……274
7.3.1 JSL 的安装……274
7.3.2 JSL 的使用……275
7.4 Dojo 对国际化（i18n）的支持……276
7.4.1 编码声明……276
7.4.2 使用本地化资源……277
7.4.3 日期、时间、数字、汇率的国际化……280
7.5 前进、后退按钮和添加收藏网址……280
7.5.1 问题的提出……280
7.5.2 Dojo 的解决方案……281
7.5.3 应用示例……282
7.5.4 dojo.io.bind 对前进、后退按钮的支持……286
7.6 自定义 Dojo Widget……289
7.6.1 扩展 Dojo 自带 Widget……289
7.6.2 定义新的 Widget……291
7.7 Dojo 开发和调试技巧……298
7.7.1 JavaScript 开发和调试工具……298
7.7.2 输出调试信息……298
7.7.3 Dojo 全局配置对象 djConfig……300
7.8 小结……300

第 8 章 Dojo 应用实例——电子邮件系统……301
8.1 应用背景……301
8.2 需求分析……301
8.3 系统设计与实现……302
8.3.1 邮件服务器的搭建……302

8.3.2 数据库设计 ····· 307
8.3.3 服务器端接口 ····· 309
8.3.4 系统实现 ····· 311
8.4 小结 ····· 338

附录 Ajax 开发工具集 ····· 339
A.1 Venkman ····· 339
A.2 Microsoft 系列 JavaScript 调试器 ····· 341
A.3 FireBug ····· 343
A.4 Web Developer ····· 345
A.5 IE DOM Inspector ····· 346
A.6 Aptana ····· 346
A.7 Ajax Toolkit Framework ····· 347
A.8 JsEclipse ····· 348
A.9 Fiddler ····· 349
A.10 Tamper Data ····· 349



第 1 章　Ajax 基础

学习 Ajax 首先要先弄清楚它可以做什么。目前，应用程序可以分为两大类：C/S（客户端/服务器）结构和 B/S（浏览器/服务器）结构。C/S 结构的应用程序一般响应速度较快，具有漂亮的用户界面和优良的动态性，但是 C/S 应用程序需要将可执行文件安装到客户机上，程序的部署及其更新的复杂性是一直困扰用户和开发人员的问题。B/S 应用程序的最大优点在于它无须在客户端安装，客户端只需要浏览器就可以运行，程序的部署及其更新也相对简单，但是 B/S 结构的应用程序缺乏良好的交互性，应用模式比较单一，即请求－等待－请求，用户体验与 C/S 应用无法相比。而 Ajax 技术的出现，正在改变 B/S 结构的应用模式，它使得 B/S 应用也能够实现 C/S 应用的功能和交互性，用户界面更加友好。

Ajax 最典型的应用当属 Google 的 Web 邮件系统——Gmail。与传统的 Web 邮件系统相比，Gmail 的革新在于它无处不体现了以用户为先的原则，无论是阅读邮件、撰写回复邮件，还是对已有邮件的管理方式，都非常方便。可以说 Ajax 是实现这些良好用户体验的技术基础，而 Gmail 邮件系统也将 Ajax 应用到了一个相当的高度。本章首先会简要介绍什么是 Ajax、Ajax 的基本原理是什么，接下来对 Ajax 开发需要注意的问题、关键技术进行详细地讲解，最后对常见的主流 Ajax 开发框架进行介绍。

1.1　Ajax 简介

Ajax 是异步 JavaScript 和 XML（Asynchronous JavaScript and XML）的英文缩写。确切地说，Ajax 不是一种技术，而是将一系列相关技术组合应用的技巧，这些技术包括：

- 使用 XHTML（可扩展超文本标记语言，Extensible HyperText Markup Language）和 CSS（层叠样式表，Cascading Style Sheet）编写结构化的 Web 页面；
- 使用 DOM（文档对象模型，Document Object Model）进行动态显示和交互；
- 使用 XML（可扩展标识语言，Extensible Markup Language）和 XSLT（可扩展样式表语言转换，Extensible Stylesheet Language Transformation）进行数据交互和操作；

- 使用 XMLHttpRequest 进行异步数据接收；
- 使用 JavaScript 将它们绑定在一起。

“老技术，新技巧”是对 Ajax 比较恰如其分的描述。实际上早在 1998 年，微软公司的 Web 版 Outlook（Ootlook Web Access）就已经实现了类似桌面应用程序的 Web 应用。而 Ajax 在近年来如此火热，是与 Google 公司密不可分的。Google 推出的一些服务，包括 Gmail、Google Maps、Google Notebook 等，让人们切实感受到 Ajax 的独特魅力。

众所周知，在传统的 Web 应用程序中，用户在页面上填写表单字段，然后单击“提交”按钮，整个表单会被发送到服务器端，服务器将它转发给表单处理程序，表单处理完成后再发送回全新的页面。相应的应用程序模型示意图如图 1-1 所示。

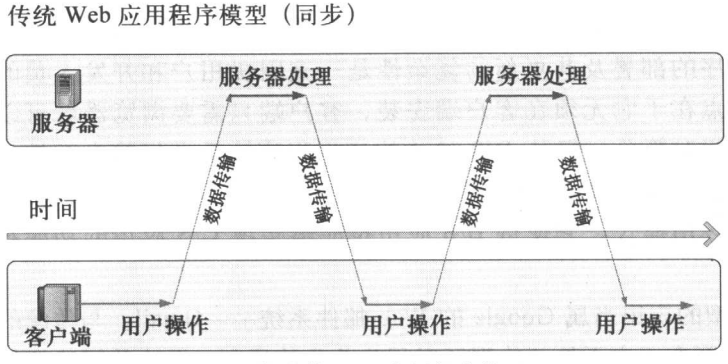


图 1-1　传统的 Web 应用程序模型

使用传统的 Web 应用程序，在服务器上的脚本和程序处理表单并返回结果的时候，用户必须等待，这时浏览器显示的是一片空白，直到服务器返回数据后才会重新绘制，这就是 Web 应用交互性差的原因。因为用户得不到立即的反馈，所以感觉不同于桌面应用程序。

Ajax 通过 JavaScript 和 XMLHttpRequest 对象在 Web 表单与服务器之间建立了一个中间层，当用户填写表单时，数据将会被这个中间层接收，JavaScript 代码会捕获表单的数据，然后通过 XMLHttpRequest 对象向服务器端发送。发送的同时，用户浏览器端显示的内容不会闪烁、消失或者延迟，数据的发送和接收在后台完成。此外，由于请求是异步发送的，用户不用等待服务器的响应，可以继续在当前界面输入数据，正常使用应用程序。基于 Ajax 的应用程序模型如图 1-2 所示。

可以看出，在 Ajax 应用中用户的操作与服务器端的处理并不是同步的，这也是 Ajax 中 Asynchronous（异步）的含义。这种处理机制打破了传统的“提交－等待－提交”的 Web 应用模式，服务器处理提交数据的同时，客户端无需等待，不会出现“白屏”。更重要的是，Ajax 可以实现真正意义上的“按需取数据”，局部更新页面，从而提高了应用程序的效率，

节约了网络资源。

Ajax Web 应用程序模型（异步）

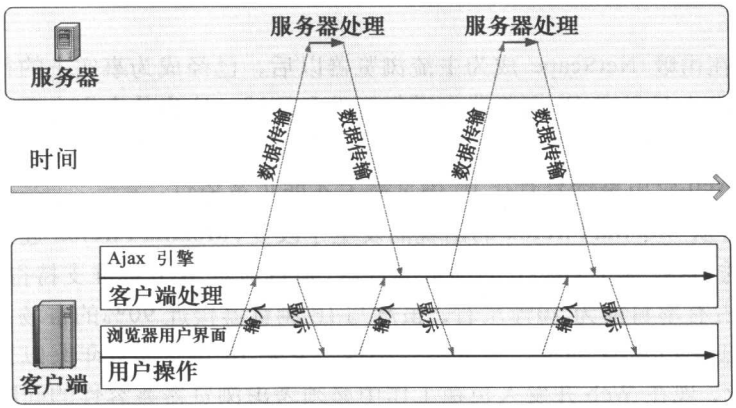


图 1-2 基于 Ajax 的 Web 应用程序模型

1.2 Ajax 开发需注意的几个问题

前面提到了 Ajax 应用的很多优点，那么如何开发基于 Ajax 的 Web 应用呢？首先，Ajax 的核心是使用了 XMLHttpRequest 对象访问服务器端资源，客户端的应用需要通过 JavaScript 语言实现，动态改变页面则是基于 DOM，因此 XMLHttpRequest、JavaScript 和 DOM 构成了 Ajax 开发的基石。

在开发 Ajax 应用时，需要特别注意以下几个问题。

1.2.1 浏览器兼容性问题

XMLHttpRequest 对象的实现需要浏览器的支持，微软公司的 Internet Exporler（IE）通过 ActiveX 控件的方式提供对 XMLHttpRequest 对象的支持，而 Mozilla 等其他浏览器则是通过创建自有的继承 XML 代理类来实现 XMLHttpRequest 对象，所以在创建 XMLHttpRequest 对象时必须判断浏览器的类型，才能使 Ajax 应用兼容各种浏览器。

不同的浏览器对于 JavaScript 语言的支持程度也不尽相同，这要从 JavaScript 的历史说起。JavaScript 作为一种脚本语言，已经流行了多年。这种脚本语言诞生自 Netscape 公司 Web 浏览器的“LiveScript”，它处于浏览器的环境中并且控制着浏览器的显示。在加入了 Java Applet 功能后，Netscape 公司决定把 LiveScript 更名为 JavaScript。微软公司在与 Netscape 浏览器竞争的时候，发明了自己的脚本语言，称为 JScript。JScript 的第一个技术标准被称为 ECMA-262，因为原本是想提供一个通用的 API，所以正规地说，JScript 应该称为 ECMAScript。ECMA 是欧洲计算机制造商协会（European Computer Manufacturers Association）的缩写，它是一个

标准化组织。为了增强 IE 浏览器的竞争优势，JScript 还实现了很多非 ECMA 标准的特性，这些特性后来被大量使用。然而使用这些非标准特性的后果就是 Web 应用的部分或者全部功能只能在 IE 浏览器上使用，这是造成 JavaScript 一直到今天都不能在浏览器之间兼容的原因之一。

IE 浏览器在击败 NetScape 成为主流浏览器以后，已经成为事实上的行业标准，有相当一部分 Web 开发人员只在 IE 浏览器上进行开发和测试，这也是人们会看到很多网站都注明“请使用 Internet Explore 5.5 以上版本访问”字样的原因。实际上，即使没有这样的说明，也有相当数量的 Web 应用系统只有在 IE 浏览器上才能正常运行。

这种情况自从 Mozilla Firefox 的出现而发生了改变，Mozilla Firefox 近几年来成为浏览器市场的后起之秀，这款基于 Gecko 的网页浏览器，是目前对 Web 标准支持得最完美的浏览器。Firefox 的市场占有率目前为 10%左右，虽然与 IE 浏览器接近 90%的市场占有率相比，仍有很大的差距，但是 Firefox 的推广速度不容小视，它是 IE 浏览器近年来以及不久的将来最强劲的对手。因此，现在 Web 开发人员在工作中必须考虑浏览器兼容性的问题，至少需要在 IE 浏览器和 Firefox 上通过测试。

Ajax 应用需要编写大量的 JavaScript 实现，如何编写兼容各个浏览器的 JavaScript 代码，是一个困难且又必须注意的问题。

此外，DOM/CSS 的支持程度在不同的浏览器之间也存在着很大的差别，这也是开发 Ajax 应用必须注意的问题。Ajax 应用中有很多的网页特效和行为是基于 DOM 和 CSS 编写的，开发人员必须考虑 DOM/CSS 在不同浏览器上的表现差异。

1.2.2 XMLHttpRequest 对象封装

前面提到，在不同浏览器中 XMLHttpRequest 对象的实现是不同的，因此在 JavaScript 中创建 XMLHttpRequest 对象时，经常会采用如下的代码：

```
/* Create a new XMLHttpRequest object to talk to the Web server */
var xmlHttp = false;
/*@cc_on @*/
/*@if (@_jscript_version >= 5)
try {
  xmlHttp = new ActiveXObject("Msxml2.XMLHTTP");
} catch (e) {
  try {
    xmlHttp = new ActiveXObject("Microsoft.XMLHTTP");
  } catch (e2) {
    xmlHttp = false;
  }
}
@end @*/

if (!xmlHttp && typeof XMLHttpRequest != 'undefined') {
```

```
  xmlHttp = new XMLHttpRequest();
}
```

这段代码的核心分为3个步骤。

- 建立一个变量xmlHttp来引用即将创建的XMLHttpRequest对象。
- 尝试在微软公司浏览器中创建XMLHttpRequest对象：
  - 尝试使用Msxml2.XMLHTTP对象创建它；
  - 如果失败，再尝试使用Microsoft.XMLHTTP对象创建它。
- 如果仍然没有建立xmlHttp，则以非Microsoft的方式创建该对象。

作为Ajax应用的核心，XMLHttpRequest对象会在Web页面的JavaScript代码中频繁使用，这样的代码也会经常出现在Web页面中。当Ajax应用的规模逐渐扩大，必然会造成大量的冗余代码，因此为了方便调用，封装XMLHttpRequest对象是十分必要的。

例如如下的代码就实现了对XMLHttpRequest对象的封装。

```
function CallBackObject() {
    this.XmlHttp = this.GetHttpObject();
}

CallBackObject.prototype.GetHttpObject = function() {
    var xmlhttp;

  /*@cc_on
  @if (@_jscript_version >= 5)
    try {
      xmlhttp = new ActiveXObject("Msxml2.XMLHTTP");
    } catch (e) {
      try {
        xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
      } catch (E) {
        xmlhttp = false;
      }
    }
  @else
  xmlhttp = false;
  @end @*/

  if (!xmlhttp && typeof XMLHttpRequest != 'undefined') {
    try {
      xmlhttp = new XMLHttpRequest();
    } catch (e) {
      xmlhttp = false;
    }
  }
  return xmlhttp;
}
```

```
CallBackObject.prototype.DoCallBack = function(URL) {
  if (this.XmlHttp) {
    if (this.XmlHttp.readyState == 4 || this.XmlHttp.readyState == 0) {
      var oThis = this;
      this.XmlHttp.open('POST', URL);
      this.XmlHttp.onreadystatechange = function() {
          oThis.ReadyStateChange();
      };
      this.XmlHttp.setRequestHeader('Content-Type',
          'application/x-www-form-urlencoded');
      this.XmlHttp.send(null);
    }
  }
}

CallBackObject.prototype.AbortCallBack = function() {
  if (this.XmlHttp)
    this.XmlHttp.abort();
}

CallBackObject.prototype.OnLoading = function() {
  // Loading
}

CallBackObject.prototype.OnLoaded = function() {
  // Loaded
}

CallBackObject.prototype.OnInteractive = function() {
  // Interactive
}

CallBackObject.prototype.OnComplete = function(responseText, responseXml) {
  // Complete
}

CallBackObject.prototype.OnAbort = function() {
  // Abort
}

CallBackObject.prototype.OnError = function(status, statusText) {
  // Error
}

CallBackObject.prototype.ReadyStateChange = function() {
  if (this.XmlHttp.readyState == 1) {
    this.OnLoading();
  } else if (this.XmlHttp.readyState == 2) {
      this.OnLoaded();
```