

JavaScript著名专家撰写

指导读者进入JavaScript框架设计的**魔法指南**

JavaScript 框架设计

司徒正美 ◎ 编著



 人民邮电出版社
POSTS & TELECOM PRESS



014035117

TP312JA
1571

JavaScript 框架设计

司徒正美 © 编著



TP312JA
1571



北航 C1714615

人民邮电出版社
北京

01032117

图书在版编目 (CIP) 数据

JavaScript 框架设计 / 司徒正美编著. -- 北京 : 人民邮电出版社, 2014. 4
ISBN 978-7-115-34358-1

I. ①J… II. ①司… III. ①JAVA语言—程序设计
IV. ①TP312

中国版本图书馆CIP数据核字(2013)第316783号

内 容 提 要

本书是一本全面讲解 JavaScript 框架设计的图书,详细地讲解了设计框架需要具备的知识,主要包括的内容为:框架与库、JavaScript 框架分类、JavaScript 框架的主要功能、种子模块、模块加载系统、语言模块、浏览器嗅探与特征侦测、样式的支持侦测、类工厂、JavaScript 对类的支撑、选择器引擎、浏览器内置的寻找元素的方法、节点模块、一些有趣的元素节点、数据缓存系统、样式模块、个别样式的特殊处理、属性模块、jQuery 的属性系统、事件系统、异步处理、JavaScript 异步处理的前景、数据交互模块、一个完整的 Ajax 实现、动画引擎、API 的设计、插件化、当前主流 MVVM 框架介绍、监控数组与子模板等。

本书适合前端设计人员、JavaScript 开发者、移动 UI 设计者、程序员和项目经理阅读,也可作为大中专院校相关专业的师生学习用书和培训学校的教材。

-
- ◆ 编 著 司徒正美
 责任编辑 张 涛
 责任印制 程彦红 杨林杰
 - ◆ 人民邮电出版社出版发行 北京市丰台区成寿寺路 11 号
 邮编 100164 电子邮件 315@ptpress.com.cn
 网址 <http://www.ptpress.com.cn>
 北京天宇星印刷厂印刷
 - ◆ 开本: 800×1000 1/16
 印张: 28.75
 字数: 701 千字
 印数: 1-3 500 册
- 2014 年 4 月第 1 版
2014 年 4 月北京第 1 次印刷
-

定价: 89.00 元

读者服务热线: (010)81055410 印装质量热线: (010)81055316

反盗版热线: (010)81055315

广告经营许可证: 京崇工商广字第 0021 号

框架则是一个半成品的应用，直接给出一个骨架，还例如盖房子，照着图纸砌砖、铺地板与涂漆就行了。在后端 Java 的三大框架中，程序员基本上就是与 XML 打交道，用配置就可以处理 80% 的编程问题。

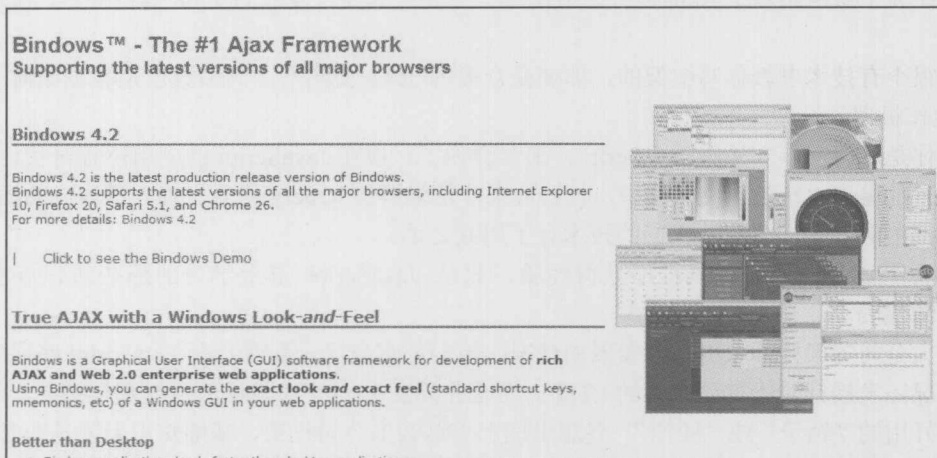
从上面描述来看，框架带来的便利性无疑比库好许多。但前端 JavaScript 由于存在一个远程加载的问题，对 JavaScript 文件的体积限制很大，因此，框架在几年前都不怎么吃香。现在网速快多了，设计师在网页制造的地位（UED）也不比昔日，因此，集成程度更高的 MVC、MVVM 框架也相继面世。

不过，无论是框架还是库，只要在浏览器中运行，就要与 DOM 打交道。如果不用 jQuery，就要发明一个半成品 jQuery 或一个半成品 Prototype。对想提升自己能力的人来说，答案其实很明显，写框架还能提升自己的架构能力。

2. JavaScript 发展历程

第 1 时期，洪荒时代。从 1995 年到 2005 年，就是从 JavaScript 发明到 Ajax 概念^①的提出。其间打了第一场浏览器战争，IE VS Netscape。这两者的 DOM API 出入很大，前端开发人员被迫改进技术，为了不想兼容某一个浏览器，发明 UA（navigator.userAgent）嗅探技术。

这个时期的杰出代表是 Bindows^②，2003 年发布，它提供了一个完整的 Windows 桌面系统，支持能在 EXT 看到的各种控件，如菜单、树、表格、滑动条、切换卡、弹出层、测量仪表（使用 VML 实现，现在又支持 SVG）。现在版本号是 4.x，如下图所示。



The image shows a screenshot of the Bindows website. The main heading is "Bindows™ - The #1 Ajax Framework" with the subtext "Supporting the latest versions of all major browsers". Below this, there is a section for "Bindows 4.2" which states it is the latest production release version and supports Internet Explorer 10, Firefox 20, Safari 5.1, and Chrome 26. A link is provided to "Click to see the Bindows Demo". Another section titled "True AJAX with a Windows Look-and-Feel" describes Bindows as a GUI software framework for rich AJAX and Web 2.0 enterprise web applications, highlighting its ability to generate a Windows-like look and feel. The bottom section is titled "Better than Desktop". On the right side of the screenshot, there is a collage of various web application interfaces that mimic the Windows desktop environment, including windows, menus, and toolbars.

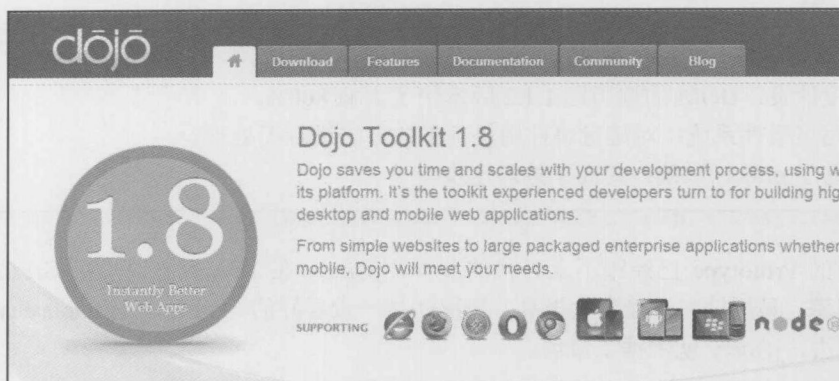
其他比较著名的还有 Dojo（2004 年）、Sarissa（2003 年）、JavaScript Remote Scripting（2000 年）。

Dojo 有 IBM 做后台，有庞大的开发团队在做，质量有保证，被广泛整合到各大 Java 框架内（struct2、Tapestry、Eclipse ATF、MyFaces）。特点是功能无所不包，主要分为 Core、Dijit、DojoX 3 大块。Core 提供 Ajax、events、packaging、CSS-based querying、animations、JSON 等相关操作 API。

^① <http://www.adaptivepath.com/ideas/ajax-new-approach-web-applications>

^② <http://www.bindows.net/>

Dijit 是一个可更换皮肤、基于模板的 Web UI 控件库。DojoX 包括一些新颖的代码和控件，如 DateGrid、charts、离线应用和跨浏览器矢量绘图等，如下图所示。



JavaScript Remote Scripting 是较经典的远程脚本访问组件，支持将客户端数据通过服务器做代理进行远程的数据/操作交互。

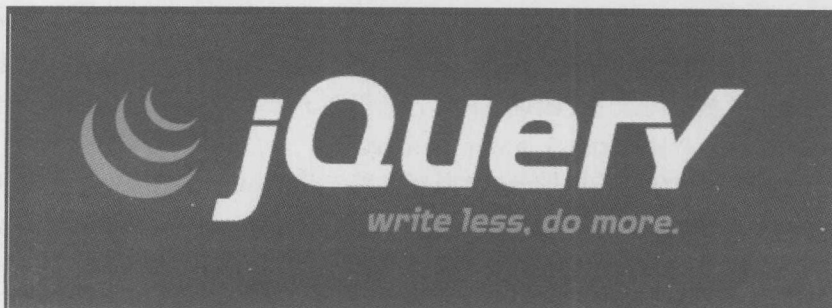
Sarissa 封装了在浏览器端独立调用 XML 的功能。

第 2 时期，Prototype“王朝”，2005 年~2008 年。其间打了第 2 次浏览器战争，交战双方是 IE6、IE7、IE8 VS Firefox 1、Firefox 2、Firefox 3，最后 Firefox 3 大胜。浏览器战争中，Prototype 积极进行升级，加入诸多 DOM 功能，因此，Jser（JavaScript 程序员）比之前好过多了。加之有 rails、script.aculo.us（一流的特效库）、Rico 等助阵，迅速占领了市场。

Prototype 时期，面向对象技术发展到极致，许多组件成套推出。DOM 特征发掘也有序进行，再也不依靠浏览器嗅探去刻意屏蔽某一个浏览器了。无侵入式 JavaScript 开发得到推崇，所有 JavaScript 代码都抽离到 JavaScript 文件，不在标签内“兴风作浪”了。

Prototype 的发展无可限量，直到 1.5 版本对 DOM 进行处理，这是一个错误^①。如它一个很好用的 API-getElementsByClassName，由于 W3C 的标准化，Prototype 升级慢了，它对 DOM 的扩展成为了它的“地雷”。

第 3 时期，jQuery 纪元，2008 年到现在（如下图所示）。



^① 详见 Prototype 核心成员的反思：<http://perfectionkills.com/whats-wrong-with-extending-the-dom/>

jQuery 则以包裹方式来处理 DOM，而且配合它的选择器引擎，若一下子选到 N 个元素，那么就处理 N 个元素，是集化操作，与主流的方式完全不一样。此外，它的方法名都起得很特别，人们一时很难接受。

2007 年 7 月 1 日，jQuery 发布了 1.1.3 版本，它的宣传是。

- (1) 速度改良：DOM 的遍历比 1.1.2 版本快了大概 800%。
- (2) 重写了事件系统：对键盘事件用更优雅的方法进行了处理。
- (3) 重写了 effects 系统：提高了处理速度。

停滞不前的 Prototype 已经跟不上时代的节奏，jQuery 在 1.3x 版本时更换 Sizzle，更纯净的 CSS 选择器引擎，易用性与性能大大提高，被程序员一致看好的 mouseenter、mouseleave 及事件代理也被整合进去，jQuery 就占据了市场。

3. JavaScript 框架分类

如果是从内部架构与理念划分，目前 JavaScript 框架可以划分为 5 类。

第 1 种出现的是以命名空间为导向的类库或框架，如创建一个数组用 `new Array()`，生成一个对象用 `new Object()`，完全的 Java 风格，因此我们就可以以某一对象为根，不断为它添加对象属性或二级对象属性来组织代码，金字塔般地垒叠起来。代表作如早期的 YUI 与 EXT。

第 2 种出现的是以类工厂为导向的框架，如著名的 Prototype，还有 mootools、Base2、Ten。它们基本上除了最基本的命名空间，其他模块都是一个由类工厂衍生出来的类对象。尤其是 mootools 1.3 把所有类型都封装成 Type 类型。

第 3 种就是以 jQuery 为代表的以选择器为导向的框架，整个框架或库主体是一个特殊类数组对象，方便集化操作——因为选择器通常是一下子选择到 N 个元素节点，于是便一并处理了。jQuery 包含了几样了不起的东西：“无 new 实例化”技术，`$(expr)` 就是返回一个实例，不需要显式地 new 出来；`get first set all` 访问规则；数据缓存系统。这样就可以复制节点的事件了。此外，IIFE (Immediately-Invoked Function Expression) 也被发掘出来。

第 4 种就是以加载器串联起来的框架，它们都有复数个 JavaScript 文件，每个 JavaScript 文件都以固定规则编写。其中最著名的莫过于 AMD。模块化是 JavaScript 走向工业化的标志。《Unix 编程艺术》列举的众多“金科玉律”的第一条就是模块，里面有言——“要编写复杂软件又不至于一败涂地的唯一方法，就是用定义清晰的接口把若干简单模块组合起来，如此一来，多数问题只会出现在局部，那么还有希望对局部进行改进或优化，而不至于牵动全身”。许多企业内部框架都基本采取这种架构，如 Dojo、YUI、kissy、qwrap 和 mass 等。

第 5 种就是具有明确分层构架的 MV* 框架。首先是 JavaScript MVC (现在叫 CanJS)、backbonejs 和 spinejs，然后更符合前端实际的 MVVM 框架，如 knockout、ember、angular、avalon、winjs。在 MVVM 框架中，原有 DOM 操作被声明式绑定取代了，由框架自行处理，用户只需专注于业务代码。

4. JavaScript 框架的主要功能

下面先看看主流框架有什么功能。这里面包含 jQuery 这个自称为库的东西，但它接近 9000 行，功能比 Prototype 还齐备。这些框架类库的模块划分主要依据它们在 github 中的源代码，基本上都是一个模块一个 JavaScript 文件。

jQuery

jQuery 强在它专注于 DOM 操作的思路一开始就是对的，以后就是不断在兼容性、性能上进行改进。

jQuery 经过多年的发展，拥有庞大的插件与完善的 Bug 提交渠道，因此，可以通过社区的力量不断完善自身。

Prototype.js

早期的王者，它分为 4 大部分。

- 语言扩展。
- DOM 扩展。
- Ajax 部分。
- 废弃部分（新版本使用其他方法实现原有功能）。

Prototype.js 的语言扩展覆盖面非常广，包括所有基本数据类型及从语言借鉴过来的“类”。其中 Enumerable 只是一个普通的方法包，ObjectRange、PeriodicalExecuter、Templat 则是用 Class 类工厂生产出来的。Class 类工厂来自社区贡献。

mootools

它由于 API 设计得非常优雅，其官方网站上有许多优质插件，因此才没有在原型扩展的反对浪潮中没落。

RightJS

又一个在原型上进行扩展的框架。

MochiKit

一个 Python 风格的框架，以前能进世界前十名的。

Ten

日本著名博客社区 Hatena 的 JavaScript 框架，由 amachang 开发，受 Prototype.js 影响，是最早以命名空间为导向的框架的典范。

mass Framework

它是一个模块化，以大模块开发为目标，jQuery 式的框架。

经过细节比较，我们很容易得出以下框架特征的结论。

- 对基本数据类型的操作是基础，如 jQuery 就提供了 trim、camelCase、each、map 等方法，Prototype.js 等侵入式框架则在原型上添加 camelize 等方法。
- 类型的判定必不可少，常见形式是 isXXX 系列。
- 选择器、domReady、Ajax 是现代框架的标配。
- DOM 操作是重中之重，节点的遍历、样式操作、属性操作也属于它的范畴，是否细分就看

框架的规模了。

- browser sniff 已过时，feature detect 正被应用。不过特性侦测还是有局限性，如果针对于某个浏览器版本的渲染 Bug、安全策略或某些 Bug 的修正，还是要用到浏览器嗅探。但它应该独立成一个模块或插件，移出框架的核心。
- 现在主流的事件系统都支持事件代理。
- 数据的缓存与处理，目前浏览器也提供 data-* 属性进行这面的工作，但不太好用，需要框架的进一步封装。
- 动画引擎，除非你的框架像 Prototype.js 那样拥有像 script.aculo.us 这样顶级的动画框架做后盾，最好也加上。
- 插件的易开发和扩展性。
- 提供诸如 Deferred 这样处理异步的解决方案。
- 即使不专门提供一个类工厂，也应该存在一个名为 extend 或 mixin 的方法对对象进行扩展。jQuery 虽然没有类工厂，但在 jQuery UI 中也不得不增加一个，可见其重要性。
- 自从 jQuery 出来一个名为 noConflict 的方法，新兴的框架都带此方法，以求狭缝中生存。
- 许多框架非常重视 Cookie 操作。

最后感谢一下业内一些朋友的帮忙，要不是他们，书不会这么顺利地写出来。以下排名不分先后：玉伯、汤姆大叔、弹窗教主、獏大、linxz、正则帝 abcd。这些都是专家级人物，在业界早已闻名遐迩。由于本人水平有限，书中难免存有不妥之处，请读者批评指正，源程序和答疑网址：<https://github.com/RubyLouvre/jsbook/issues>。编辑联系邮箱：zhangtao@ptpress.com.cn。

目 录

第 1 章 种子模块	1	5.2 各种类工厂的实现	77
1.1 命名空间	1	5.2.1 相当精巧的库——P.js	77
1.2 对象扩展	3	5.2.2 JS.Class	80
1.3 数组化	4	5.2.3 simple-inheritance	82
1.4 类型的判定	6	5.2.4 体现 JavaScript 灵活性的 库——def.js	84
1.5 主流框架引入的机制—— domReady	14	5.3 es5 属性描述符对 OO 库的冲击	88
1.6 无冲突处理	16	第 6 章 选择器引擎	100
第 2 章 模块加载系统	18	6.1 浏览器内置的寻找元素的方法	100
2.1 AMD 规范	18	6.2 getElementBySelector	102
2.2 加载器所在路径的探知	19	6.3 选择器引擎涉及的知识点	106
2.3 require 方法	21	6.4 选择器引擎涉及的通用函数	114
2.4 define 方法	27	6.4.1 isXML	114
第 3 章 语言模块	31	6.4.2 contains	115
3.1 字符串的扩展与修复	31	6.4.3 节点排序与去重	117
3.2 数组的扩展与修复	43	6.4.4 切割器	121
3.3 数值的扩展与修复	50	6.4.5 属性选择器对于空白 字符的匹配策略	123
3.4 函数的扩展与修复	56	6.4.6 子元素过滤伪类的分解与 匹配	125
3.5 日期的扩展与修复	61	6.5 Sizzle 引擎	127
第 4 章 浏览器嗅探与特征侦测	64	第 7 章 节点模块	137
4.1 判定浏览器	64	7.1 节点的创建	138
4.2 事件的支持侦测	67	7.2 节点的插入	149
4.3 样式的支持侦测	69	7.3 节点的复制	155
4.4 jQuery 一些常用特征的含义	70	7.4 节点的移除	158
第 5 章 类工厂	72	7.5 innerHTML、innerText 与 outerHTML 的处理	161
5.1 JavaScript 对类的支撑	72		

7.6 一些奇葩的元素节点	164	10.5 Prototype.js 的属性系统	240
7.6.1 iframe 元素	164	10.6 jQuery 的属性系统	246
7.6.2 object 元素	174	10.7 mass Framework 的属性系统	249
7.6.3 video 标签	179	10.8 value 的操作	253
第 8 章 数据缓存系统	185	第 11 章 事件系统	256
8.1 jQuery 的第 1 代缓存系统	185	11.1 onXXX 绑定方式的缺陷	257
8.2 jQuery 的第 2 代缓存系统	190	11.2 attachEvent 的缺陷	258
8.3 mass Framework 的第 1 代数据 缓存系统	193	11.3 addEventListener 的缺陷	259
8.4 mass Framework 的第 2 代数据 缓存系统	196	11.4 Dean Edward 的 addEvent.js 源码分析	260
8.5 mass Framework 的第 3 代数据 缓存系统	198	11.5 jquery1.8.2 的事件模块概览	263
8.6 总结	199	11.6 jQuery.event.add 的源码解读	266
第 9 章 样式模块	200	11.7 jQuery.event.remove 的源码 解读	269
9.1 主体结构	201	11.8 jQuery.event.dispatch 的源码 解读	271
9.2 样式名的修正	205	11.9 jQuery.event.trigger 的源码 解读	276
9.3 个别样式的特殊处理	206	11.10 jQuery 对事件对象的修复	280
9.3.1 opacity	206	11.11 滚轮事件的修复	286
9.3.2 user-select	208	11.12 mouseenter 与 mouseleave 事件的修复	290
9.3.3 background-position	208	11.13 focusin 与 focusout 事件的 修复	293
9.3.4 z-index	209	11.14 旧版本 IE 下 submit 的事件 代理的实现	295
9.3.5 盒子模型	210	11.15 oninput 事件的兼容性处理	296
9.3.6 元素的尺寸	211	第 12 章 章异步处理	298
9.3.7 元素的显隐	218	12.1 setTimeout 与 setInterval	299
9.3.8 元素的坐标	222	12.2 Mochikit Deferred	301
9.4 元素的滚动条的坐标	228	12.3 JSDDeferred	309
第 10 章 属性模块	229	12.3.1 得到一个 Deferred 实例	310
10.1 如何区分固有属性与 自定义属性	231	12.3.2 Deferred 链的实现	312
10.2 如何判定浏览器是否区分固 有属性与自定义属性	233	12.3.3 JSDDeferred 的并归 结果	316
10.3 IE 的属性系统的三次演变	234		
10.4 className 的操作	235		

12.3.4 JSDeferred 的性能 提速	318	14.4 mass Framework 基于 JavaScript 的 动画引擎	369
12.4 jQuery Deferred	321	14.5 requestAnimationFrame	377
12.5 Promise/A 与 mmDeferred	327	14.6 CSS3 transition	383
12.6 JavaScript 异步处理的前景	334	14.7 CSS3 animation	388
第 13 章 数据交互模块	339	14.8 mass Framework 基于 CSS 的 动画引擎	390
13.1 Ajax 概览	339	第 15 章 插件化	398
13.2 优雅地取得 XMLHttpRequest 对象	339	15.1 jQuery 的插件的一般写法	398
13.3 XMLHttpRequest 对象的 事件绑定与状态维护	342	15.2 jQuery UI 对内部类的操作	401
13.4 发送请求与数据	344	15.3 jQuery easy UI 的智能加载 与个别化制定	403
13.5 接收数据	346	15.4 更直接地操作 UI 实例	406
13.6 上传文件	349	第 16 章 MVVM	409
13.7 一个完整的 Ajax 实现	351	16.1 当前主流 MVVM 框架介绍	410
第 14 章 动画引擎	363	16.2 属性变化的监听	416
14.1 动画的原理	363	16.3 ViewModel	418
14.2 缓动公式	365	16.4 绑定	429
14.3 API 的设计	368	16.5 监控数组与子模板	437

第1章 种子模块

种子模块也叫核心模块，是框架的最先执行的部分。即便像 jQuery 那样的单文件函数库，它的内部也分许多模块，必然有一些模块冲在前面立即执行，有一些模块只有用到才执行，也有一些模块可有可无，存在感比较弱，只在特定浏览器下才运行。

种子模块就是其中的急先锋，它里面的方法不一定要求个个神通广大，设计优良，但一定极具扩展性，常用，稳定。扩展性是指通过它们能将其他模块的方法包进来，让种子像大树一样成长；常用是指绝大多数模块都用到它们，防止做重复工作；稳定是指不能轻易在以后版本就给去掉，要信守承诺。

参照许多框架与库的实现，我认为种子模块应该包含如下功能：对象扩展，数组化，类型判定，简单的事件绑定与卸载，无冲突处理，模块加载与 domReady。本章的讲解内容以 mass Framework 的种子模块为范本，可以到以下地址下载：

<https://github.com/RubyLouvre/mass-Framework/blob/1.4/mass.js>

1.1 命名空间

种子模块作为一个框架的最开始部分，除了负责辅建全局用的基础设施外，你有没有想到给读者一个震撼的开场呢？俗话说，好的开头是成功的一半。

时下“霸主”jQuery 就有一个很好的开头——IIFE（立即调用函数表达式），一下子吸引住读者，让读者吃一颗定心丸——既然作者的水平如此高超，还怕什么啊，直接拿来用。

IIFE 是现代 JavaScript 框架最主要的基础设施，它像细胞膜一样包裹自身，防止变量污染。但我们总得在 Windows 里设置一个立足点，这个就是命名空间。基本上我们可以把命名空间等同于框架的名字，不过对于某些框架，它们是没有统一的命名空间，如 Prototype.js, mootools。它们就是不想让你感觉到框架的存在，它的意义深透到 JavaScript、DOM、COM 等整个执行环境的每个角落，对原生对象的原型进行扩展。由于道格拉斯（JSON 作者）的极力反对，新兴的框架都在命名空间上构建了。

命名空间是干什么用呢？这里就不多说了。我们看怎么在 JavaScript 模拟命名空间。JavaScript 一切基于对象，但只有复合类型的对象才符合这要求，比如 function、regexp、object……不过最常

用的还是 `object` 与 `function`。我们往一个对象上添加一个属性，而这个属性又是一个对象，这个对象我们又可以为它添加一个对象，通过这种方法，我们就可以有条不紊地构建我们的框架。用户想调用某个方法，它就以 `XXX.YYY.ZZZ()` 的形式调用。

```

if (typeof(Ten) === "undefined") {
    Ten = {};
    Ten.Function = { /*略*/ }
    Ten.Array = { /*略*/ }
    Ten.Class = { /*略*/ }
    Ten.JSONP = new Ten.Class( /*略*/ )
    Ten.XHR = new Ten.Class( /*略*/ )
}

```

纵观各大类库的实现，一开始基本都是定义一个全局变量作为命名空间，然后对它进行扩展，如 `Base2` 的 `Base`、`Ext` 的 `Ext`、`jQuery` 的 `jQuery`、`YUI` 的 `YUI`、`dojo` 的 `dojo`、`MochiKit` 的 `MochiKit` 等。从全局变量的污染程度来看，分为两大类。

`Prototype`、`mootools` 与 `Base2` 归为一类。`Prototype` 的哲学是对 `JavaScript` 原生对象进行扩展。早些年，`Prototype` 差点成为事实的标准，因此基本没有考虑到与其他库的共存问题。基于 `Prototype`，也发展出诸如 `script.aculo.us`、`rico`、`Plotr`、`ProtoChart`、`Scripty 2` 等非常优秀的类库及一大堆收费插件，非 `jQuery` 那一大堆垃圾插件所能比拟的。而且，有点渊源的插件几乎都与 `Prototype` 有关，如著名的 `lightbox`。`mootools` 是 `Prototype` 的升级版，更加 `OO`，全面复制其 `API`。`Base` 则是想修复 `IE` 的 `Bug`，让 `IE` 拥有标准浏览器的 `API`，因此也把所有原生对象污染一遍。

第二类是 `jQuery`、`YUI`、`EXT` 这些框架。`YUI` 与 `EXT` 就是像上面给出的代码那样，以叠罗汉方式构建的。`jQuery` 则另辟蹊径，它是以选择器为导向的，因此它的命名空间是一个函数，方便用户把 `CSS` 表达器字符串传进来，然后通过选择器引擎进行查找，最后返回一个 `jQuery` 实例。另外，像 `jQuery` 最初也是非常弱小的，它想让人家试用自己的框架，但也想像 `Prototype` 那样使用美元符号作为它的命名空间。因此它特意实现了多库共存机制，在 `$`、`jQuery` 与用户指定的新命名空间中任意切换。

`jQuery` 的多库共存原理很简单，因此后来也成为许多小库的标配。首先把命名空间保存到一个临时变量中，注意这时这个对象并不是自己框架的东西，可能是 `Prototype.js` 等巨头的，然后再搞个 `noConflict` 放回去。

```

//jQuery1.2
var _jQuery = window.jQuery, _$ = window.$; //先把可能存在的同名变量保存起来

jQuery.extend({
    noConflict: function(deep) {
        window.$ = _$; //这时再放回去
        if (deep)
            window.jQuery = _jQuery;
        return jQuery;
    }
})

```

但 jQuery 的 noConflict 只对单文件的类库框架有用，像 EXT 就不能复制了。因此把命名空间改名后，将 EXT 置为 null，然后又通过动态加载方式引入新的 JavaScript 文件，该文件再以 EXT 调用，将会导致报错。

mass Framework 对 jQuery 的多库共存进行改进，它与 jQuery 一样拥有两个命名空间，一个是美元符号的短命名空间，另一个是根据 URL 动态生成的长命名空间（jQuery 就是 jQuery!）。

```
namespace = DOC.URL.replace(/(#+|\W)/g, '');
```

短的命名空间随用户改名，长的命名空间则是加载新的模块时用的，虽然用户在模块中使用 \$ 做命名空间，但当 JavaScript 文件加载下来时，我们会对里面的内容再包一层，将 \$ 指向正确的对象，具体实现见 define 方法。

12 对象扩展

我们需要一种机制，将新功能添加到我们的命名空间上。这方法在 JavaScript 通常被称做 extend 或 mixin。JavaScript 对象在属性描述符（Property Descriptor）没有诞生之前，是可以随意添加、更改、删除其成员的，因此扩展一个对象非常便捷。一个简单的扩展方法实现是这样。

```
function extend(destination, source) {
    for (var property in source)
        destination[property] = source[property];
    return destination;
}
```

不过，旧版本 IE 在这里有个问题，它认为像 Object 的原型方法就是不应该被遍历出来，因此 for in 循环是无法遍历名为 valueOf、toString 的属性名。这导致，后来人们模拟 Object.keys 方法实现时也遇到了这个问题。

```
Object.keys = Object.keys || function(obj){
    var a = [];
    for(a[a.length] in obj);
    return a ;
}
```

在不同的框架，这个方法还有不同的实现，如 EXT 分为 apply 与 applyIf 两个方法，前者会覆盖目标对象同名属性，而后者不会。dojo 允许多个对象合并在一起。jQuery 还支持深拷贝。下面是 mass Framework 的 mix 方法，支持多对象合并与选择是否覆盖。

```
function mix(target, source) {
    //如果最后参数是布尔，判定是否覆写同名属性
    var args = [].slice.call(arguments), i = 1, key,
        ride = typeof args[args.length - 1] == "boolean" ? args.pop() : true;
    if (args.length === 1) { //处理$.mix(hash) 的情形
        target = !this.window ? this : {};
        i = 0;
    }
    while ((source = args[i++])) {
        for (key in source) { //允许对象糅杂，用户保证都是对象
```

```

        if (ride || !(key in target)) {
            target[ key ] = source[ key ];
        }
    }
}
return target;
}

```

1.3 数组化

浏览器下存在许多类数组对象，如 function 内的 arguments，通过 document.forms、form.elements、document.links、select.options、document.getElementsByName、document.getElementsByTagName、childNodes、children 等方式获取的节点集合（HTMLCollection、NodeList），或依照某些特殊写法的自定义对象。

```

var arrayLike = {
    0: "a",
    1: "1",
    2: "2",
    length: 3
}

```

类数组对象是一个很好的存储结构，不过功能太弱了，为了享受纯数组的那些便捷方法，我们在处理它们前都会做一下转换。

通常来说，只要 [].slice.call 就能转换了，但旧版本 IE 下的 HTMLCollection、NodeList 不是 Object 的子类，采用如上方法将导致 IE 执行异常。我们看一下各大库怎么处理的。

```

//jQuery 的 makeArray
var makeArray = function(array) {
    var ret = [];
    if (array != null) {
        var i = array.length;
        // The window, strings (and functions) also have 'length'
        if (i == null || typeof array === "string" || jQuery.isFunction(array) ||
            array.setInterval)
            ret[0] = array;
        else
            while (i)
                ret[--i] = array[i];
    }
    return ret;
}

```

jQuery 对象是用来储存与处理 dom 元素的，它主要依赖于 setArray 方法来设置和维护长度与索引，而 setArray 的参数要求是一个数组，因此 makeArray 的地位非常重要。这方法保证就算没有参数也要返回一个空数组。

Prototype.js 的 \$A 方法：

```

function $A(iterable) {

```

```

if (!iterable)
  return [];
if (iterable.toArray)
  return iterable.toArray();
var length = iterable.length || 0, results = new Array(length);
while (length--)
  results[length] = iterable[length];
return results;
};

```

mootools 的 \$A 方法:

```

function $A(iterable) {
  if (iterable.item) {
    var l = iterable.length, array = new Array(l);
    while (l--)
      array[l] = iterable[l];
    return array;
  }
  return Array.prototype.slice.call(iterable);
};

```

Ext 的 toArray 方法:

```

var toArray = function() {
  return isIE ?
    function(a, i, j, res) {
      res = [];
      Ext.each(a, function(v) {
        res.push(v);
      });
      return res.slice(i || 0, j || res.length);
    } :
    function(a, i, j) {
      return Array.prototype.slice.call(a, i || 0, j || a.length);
    }
}();

```

Ext 的设计比较巧妙, 功能也比较强大。它一开始就自动执行自身, 以后就不用判定浏览器了。它还有两个可选参数, 对生成的纯数组进行操作。

dojo 的 `_toArray` 和 Ext 一样, 后面两个参数是可选的, 只不过第二个是偏移量, 最后一个已有的数组, 用于把新生的新组元素合并过去。

```

(function() {
  var efficient = function(obj, offset, startWith) {
    return (startWith || []).concat(Array.prototype.slice.call(obj, offset || 0));
  };

  var slow = function(obj, offset, startWith) {
    var arr = startWith || [];
    for (var x = offset || 0; x < obj.length; x++) {
      arr.push(obj[x]);
    }
    return arr;
  };
}());

```



```

};

dojo._toArray =
  dojo.isIE ? function(obj) {
    return ((obj.item) ? slow : efficient).apply(this, arguments);
  } :
    efficient;

})();

```

最后是 `mass` 的实现，与 `dojo` 一样，一开始就进行区分，W3C 方直接 `[].slice.call`，IE 自己手动实现一个 `slice` 方法。

```

$.slice = window.dispatchEvent ? function(nodes, start, end) {
  return [].slice.call(nodes, start, end);
} : function(nodes, start, end) {
  var ret = [],
      n = nodes.length;
  if (end === void 0 || typeof end === "number" && isFinite(end)) {
    start = parseInt(start, 10) || 0;
    end = end == void 0 ? n : parseInt(end, 10);
    if (start < 0) {
      start += n;
    }
    if (end > n) {
      end = n;
    }
    if (end < 0) {
      end += n;
    }
    for (var i = start; i < end; ++i) {
      ret[i - start] = nodes[i];
    }
  }
  return ret;
}

```

1.4 类型的判定

JavaScript 存在两套类型系统，一套是基本数据类型，另一套是对象类型系统。基本数据类型包括 6 种，分别是 `undefined`、`string`、`null`、`boolean`、`function`、`object`。基本数据类型是通过 `typeof` 来检测的。对象类型系统是以基础类型系统为基础的，通过 `instanceof` 来检测。然而，JavaScript 自带的这两套识别机制非常不靠谱，于是催生了 `isXXX` 系列。就拿 `typeof` 来说，它只能粗略识别出 `string`、`number`、`boolean`、`function`、`undefined`、`object` 这 6 种数据类型，无法识别 `Null`、`RegExpArgument` 等细分对象类型。

让我们看一下这里面究竟有多少陷阱。

```

typeof null // "object"
typeof document.childNodes //safari "function"
typeof document.createElement('embed')//ff3-10 "function"

```