异步图书
www.epubit.com.cn

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

（第 2 版）

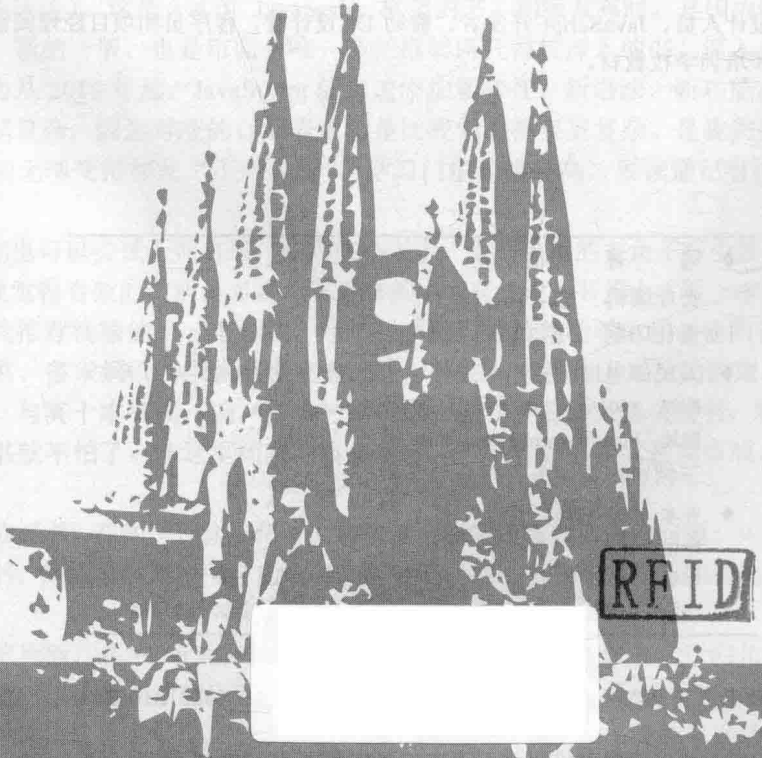
司徒正美◎编著

中国工信出版集团
人民邮电出版社
POSTS & TELECOM PRESS

JavaScript框架设计

（第2版）

司徒正美◎编著



人民邮电出版社

北京

图书在版编目（CIP）数据

JavaScript框架设计 / 司徒正美编著. -- 2版. --
北京 : 人民邮电出版社, 2017.9
ISBN 978-7-115-46429-3

Ⅰ. ①J… Ⅱ. ①司… Ⅲ. ①JAVA语言－程序设计
Ⅳ. ①TP312.8

中国版本图书馆CIP数据核字(2017)第189013号

内 容 提 要

本书全面讲解了 JavaScript 框架设计及相关的知识，主要内容包括种子模块、语言模块、浏览器嗅探与特征侦测、类工厂、选择器引擎、节点模块、数据缓存模块、样式模块、属性模块、PC 端和移动端的事件系统、jQuery 的事件系统、异步模型、数据交互模块、动画引擎、MVVM、前端模板（静态模板）、MVVM 的动态模板、性能墙与复杂墙、组件、jQuery 时代的组件方案、avalon2 的组件方案、react 的组件方案等。

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，也可作为相关专业学习用书和培训学校教材。

◆ 编　　著　司徒正美
责任编辑　张　涛
责任印制　焦志炜

◆ 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
三河市海波印务有限公司印刷

◆ 开本：800×1000　1/16
印张：29.5
字数：715 千字　　2017 年 9 月第 2 版
印数：9 001 – 12 000 册　　2017 年 9 月河北第 1 次印刷

定价：95.00 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：京东工商广登字 20170147 号

前　言

距本书前一版出版已经 3 年了，这 3 年来前端技术也发生了很大变化。因为这 3 年中，出现了团购、P2P、O2O、直播等几大创业热潮，对前端的技术需求更强，也要求更高。许多公司为了迎合用户需求，也由 PC 端转移到移动端，全新的交互方式，及之前不曾遇到的性能问题，这些都需要全新的思路与框架来解决。旧的 jQuery 已经跟不上时代的步伐，因此一些新库如雨后春笋般出来，如 fastclick、iscroll、react、fetch-polyfill、es5-shim、babel、rollup、rxjs …… 在 react 新版刚完成之际，react 的版本号已经飙升到 15，nodejs 已经发展到 8.0，这么多新东西，这么快的更新换代，一方面反映了前端技术的欣欣向荣，另一方面说明这个市场还不成熟。市场不成熟，正是框架高手“称雄一方”的好时机。远的不说，就说国内的玉伯，由于适时推出 seajs，与国外的 requirejs 竞争，国内舆论哗然，贬褒不一，但不管你说什么，时势造英雄，为什么英雄不是自己呢。说到底，实力很重要。

在前端最能反映威力的技术就是框架，其中一个评价标准是在 GitHub 上拿到很多星星的开源框架（网友的评论）。这是一本讲 JavaScript 框架的书，初版发布时，是国内唯一一本深入研究前端框架的书。新的一版，也是市面上唯一能把框架研究得较深入的书。深入并不代表难，但肯定有门槛。因为从 2015 年起，JavaScript 就加速添加新特征、新语法、新功能，框架也变得越来越庞大，越来越复杂，因为对应的行业需求总是比我们的框架更复杂。是我们的框架适应现实，不是我们的框架无端变得如此“不可理喻”，学习门槛越来越高，要读懂已有框架的难度系数越来越陡峭。

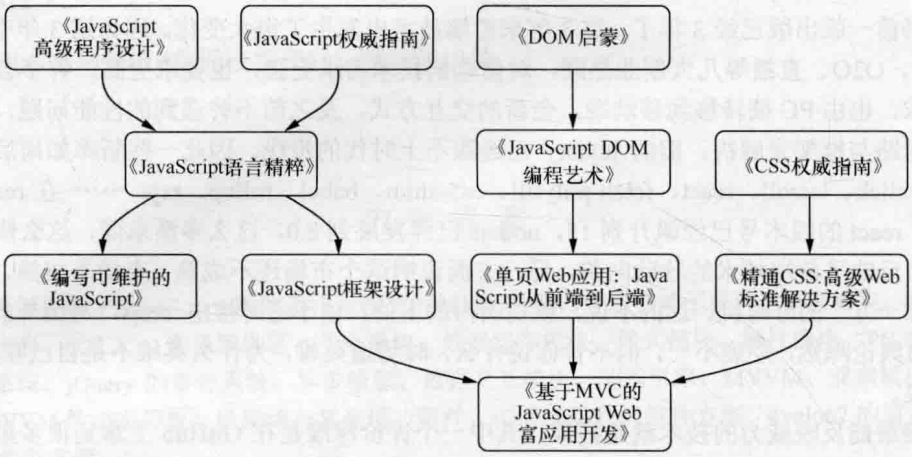
当然，你也可以尝试无师自通，在武侠小说中，无师自通的主角不在少数，但无书自通者恐怕寥寥无几。我觉得有效的方法是多看几本进阶的书。有人说，书买十本看一本也值的，我觉得有待商榷。因此我推荐钱穆的另一个说法。当年，李敖向他请教治学方法，他回答说：**并没有具体方法，要多读书、多求解，当以古书原文为底子、为主，免受他人成见的约束。书要看第一流的，一遍一遍读。与其十本书读一遍，不如一本书读十遍。不要怕大部头的书，养成读大部头书的习惯，则普通书就不怕了。**从这里面得到许多启示，一是读好书，二是读原版，三是精读，四是能读厚书。

第 1 点读好书。在知乎，总是有新人请教学习前端该看什么书的问题。其实前端的好书也有不少，如红皮书、蝴蝶书、犀牛书、乌龟书、时钟书、鸽子书……《你不知道的 JavaScript》，也很有特色。

第 2 点读原版，在 IT 行业，特指英文原版。大凡有新技术出来，它们几乎都是以英文为载体出现，然后隔一年半载，才翻译到国内，所以学习新技术还是先看原版书。

第 3 点精读，就是重复看。多看书是一定的，但如果发现书上说的东西你始终看不进去时，一定不要坚持，一般来说看不下去是因为：一、你还没有达到看本书的水平，不适合你看；二、你看

的书的确是本“烂书”，遇到这种情况赶紧换书。如果找到一本适合自己的书一定要坚持看下去，看到不懂的东西多搜索，搜索不到就多问人。当初，我阅读 jQuery 源码也是如此，最初看的是 1.4.3 版本，看得一头雾水，一气之下，从最初的 1.0 版本开始看。看完所有版本，了解其迭代过程，明白了一些代码后来为什么重构成这样。在那本书中，有相当多的源码，里面着着实实比较了程序的演化过程。



第 4 点看厚书，一些集大成者的书，比如你创作文学，就看看《文心雕龙》；研究哲学，就得看黑格尔的《哲学全书》。在 JavaScript 领域，所谓的大部头莫过于《JavaScript 权威指南》等。厚达千页，许多细节都涉及了。

看了这样的书，也不能保证你能写出漂亮的代码，更别说创造一个热门的框架。这情形就好像你学了些单词和语法，你就想听懂不带字幕的英文电影一样，那是不可能的。不同阶段，你需要掌握的技能是不一样的。框架是另一个层次的技术。但别灰心，本书就是这些知识应用的指南，从高人的博客中、浏览器的更新日志中、GitHub 的 ISSUE 中，辛辛苦苦整理来的知识，加上自己的实践分享给大家。

第一版的《JavaScript 框架设计》是完全按照一个框架的编写顺序来编排，导致阅读难度太大，让许多读者反馈说无法坚持下去。这次大改版，将一些难点挪到后面，并加上大量的流程图与示意图来降低学习难度。并且随着浏览器的升级，许多东西也该更新换代，因此本书俨然是一本新书，无论是目录结构与内容上都焕然一新。最开始的知识是偏底层，讲授一些工具方法，一些常出现在各种框架的模块，里面会大量列举实现相同功能的不同实现，比如说 isArray、trim、一个迷你的事件系统、获取元素样式、如何跨域……但越到后面，代码就越少，却能通过提供的网上链接，到相关博客里去观看相关的内容。后面几章都是非常前沿的东西，许多技术方案还没有定案（至少在我成书时），没有最佳实践，只能提供一些抽象的方法论，让读者们自行悟道。有些东西不是死记硬背就能理解的。并且，本书也不是让你造一个 avalon/angular 的框架。每个时代对框架的需求都不一样。引用《从 0 到 1》里的话——商业世界的每一刻都不会重演。下一个比尔·盖茨不会再开发操作系统，下一个拉里·佩奇或是谢尔盖·布林不会再研发搜索引擎，下一个马克·扎克伯格也

不会去创建社交网络。如果你照搬这些人的做法，你就不是向他们学习了。

最后，感谢那些帮我审稿的朋友们（以下排名不分先后）：

王方磊、曹学进、张劲松、刘可、贺文榜、方正、索平、熊小涛、周静文、余永鹏、杨忠业、刘伟、曹学进、黄建新。

还有经常伴随我的可口可乐、香飘飘奶茶、旺旺仙贝、大白兔奶糖、肉松饼、徐福记、牛轧糖、牛肉粒、鱿鱼丝、小黄鱼、曲奇饼、德芙巧克力……它们也不分先后！

本书答疑 QQ 群为（471118627），本书编辑和投稿邮箱为 zhangtao@ptpress.com.cn。

InfoQ 访谈

InfoQ：请您介绍一下自己目前的工作职责，负责的项目情况。据说您是写小说出身的，能否简单介绍一下自己的工作经历？为什么选择进入前端领域？为什么对前端框架抱有极大的热情？

笔者：现在我在“去哪儿网”任前端架构师一职，带领大家搞 react 开发。react，特别是 react native 是现时最好的移动框架。而公司的重点也转移到移动端了，因此我们小组负责将移动端相关设置全部搭建起来，各种“坑”踩一遍，努力为业务线提供最坚实的支撑。

每个人的经历都很曲折，尤其是前端人员，许多人都觉得前端比较容易学，或者赚钱比较多，才进入这行业。而我比较幸运，编代码也是我大学时的一个乐趣。因此不会像其他人，会出现动力不足的情况。当然我的兴趣还是很多的，比如写小说、建筑、考古、学日语、学动画、看科幻小说、陶艺……会找乐子的人，不会轻易泄气。

总的来说，我的兴趣都有个特点，就是缺少与他人的互动。编代码也一样，一个人静静地编就好了。像 PM，需要与别人不断沟通，我可能做不过来。每个人的性格都不一样，因此选择行业要符合自己的性格。

至于为什么进入前端，纯粹是偶然。我的一个弟弟是做这行的。我在小县城呆着赚不了几个钱，他说带我去深圳见识见识，便一下子介绍到他公司做前端了。我在小县城时也用前端接些小活过日子，因此不会觉得一下子跳跃太大。更重要的是，我特别能写程序，我没进入这行时，就写了三百多篇有关前端的博文，那时大家都以为我在大公司任职了。

出于这样的误解，公司一开始就把一些很重要的事让我做，我要确保代码的质量，因为我的组件是被许多人复用的，就从那时起，我就一直搞框架、搞组件、搞各种工具。

infoQ：请您跟大家讲一下前端框架的发展历史、前端框架的起源和发展如何？现在的前端框架很多，其背后的原因是什么？国内的前端框架又是怎样起步的、发展现状如何？

笔者：这是一个老生常谈的话题，基本每本 JavaScript 书都会聊一下这个话题。主要原因是，JavaScript 没有自己的 SDK（核心库），需要依赖“民间”的力量。最开始是一些大公司有能力开发这些框架，如 Prototype.js，也是作为 ROR 的次要项目开发出来的。此外还有 dojo、closure、YUI 这巨大的框架，也是大公司搞的。后来突然出现 jQuery 这样由天才开发的框架。事实证明，大公司那一套管理方式，以 KPI 驱动的框架有着致命的缺憾，虽然面面俱到，但不能迅速吸收 IT 社区的新东西，使用起来不够方便灵活。

再后来，大家都知道是 jQuery 的天下，大家都争先恐后地为它做插件。jQuery 也大大解放了程序员的生产力，让我们有时间做一些更有意义的事。在后 jQuery 时代，最有意义的两件事是 RequireJS 的诞生与 nodejs 的出世。前者试图解决 JavaScript 模块化问题，后者让我们能从后端那里“抢”走一些活儿，那些活儿本来就是前端做比较合适。比如说做模板、套模板、传数据、JavaScript 的语法检测、风格检测、理点等。

这个时间里，产生了像 backbone 这样的 MVC 框架。但立即被 knockout、angular、react 等 MVVM 框架占去份额了。要知道，后端从 MVC 进化到 MVVM，需要大概 10 年时间，而前端则不到 2 年。

前端框架发展实在太迅猛了！

我想其背后最大的动力是需求！源源不断的需求！原来由后端做的活儿，放在前端做更合适、更快，用户体验更好。这是趋势使然，挡也挡不住。

目前国内的发展历程其实与国外一模一样。最开始是公司牵头，后来就涌现大量出色的个人项目。阿里的前端技术之所以这么强，是因为他们不断地研制自己的“轮子”，“轮子”会越造越好。那些绝不重复造“轮子”的人默默无名，而框架作者们则开创自己一片新天地了。中国拥有世界上最庞大的互联网市场，面临的挑战很大，光是满足国内用户的需求，国内框架的研发人员就需要比国外同行更加努力。经过这几年的磨练，国内的框架也渐渐走出国门（如我的 avalon，在澳大利亚、德国都有人在用，又如百度的 ECharts，这个也非常抢眼）。

infoQ：avalon 的起源与发展是如何的？avalon 2 的架构如何？采用这样的架构有什么好处？与其他框架相比，avalon 更加“接地气”的点体现在哪些地方？

笔者：avalon 当初只是我另一个早期的框架 mass Framework 的一个插件。mass Framework 类似于 jQuery 与 Prototypejs 的结合体。没什么特色，被埋没也是必然的。但我说过，“轮子”会越造越好的。当我将这个插件介绍到**博客园**——国内一个非常著名的 IT 社区，反映很不错。于是独立出来搞。经过 5 年的发展，它渐渐拥有自己的论坛与社区。不过，由于年龄的增长，我也开始抗拒一些新东西（比如说社区上一些自动化工具，总是想自己全部实现），导致 avalon 一度发展缓慢。发展到 1.5 版时试图奋起直追，效果不明显。avalon2 决定使用一个更吸引眼球的东西扭转局面——这就是虚拟 DOM，带来了性能上的飞跃。MVVM 虽然非常方便，但很容易出现性能瓶颈。出自于谷歌之手的 angular，也有 2000 个指令（即一个页面超过 2000 个指令，页面更新就慢得令人发指）。Facebook 的 react 带来了“虚拟 DOM”这个新概念，使用轻量对象代替重型对象来承担绝大多数的页面重绘工作，解决了所谓的“性能墙”问题。

原来 MVVM 架构分 3 层，M、V、VM 三层，我们只需要关注 VM。VM 通过各种手段得知外界对它的操作，然后它智能地通知 M 与 V 进行变更。VM 承受太多职责，导致不堪重负。而虚拟 DOM 的导入，让 avalon2 拥有 4 层架构。虚拟 DOM 位于 V 与 VM 之间，复杂的视图计算由虚拟 DOM 计算好，然后 diff 出差异点实现最小化刷新。这是算法的伟大胜利。为了实现虚拟 DOM，前端框架作者也接触编译原理等高深的东西了。

现在主流的 MVVM 也结合虚拟 DOM 进行性能优化。基本上它们是基于 Object.defineProperty 这个 API。而这个 API 在 IE8 中有 bug，只能用于 IE9＋。因此它们的兼容性都比较差。而 avalon 的优势在于其作者精通各种兼容性问题与“魔法”。在 IE6～IE8 下，我找到了 VBScript 实现对 VM 的自省机制，在较新的浏览器使用 Object.defineProperty，在更新锐的浏览器，则使用 Proxy（动态代理）这个划时代的东西，从此我们可以动态监听对象是否被添加删除了某个属性，或调用了某个方法，而不像 Object.defineProperty 只能监听读写操作（Proxy 对象被用于定义自定义基本操作的行为，如属性查找、分配、枚举、函数调用等，在 Firefox 的定义中一共有 14 个属性）。

从上面的描述来看，avalon 走在时代的前列，但它不忘初心，还继续支持 IE6，让大家用 MVVM 或虚拟 DOM 时没有后顾之忧。并且大家也不用担心 avalon 为了兼容 IE6，会变得非常冗长。因为 avalon 是一份源码编译出好几个版本，每个版本根据浏览器的支持程度合并对应的模板。如果只想运行于 IE10，其会相当小。

infoQ：在选择前端框架时，大家的建议很多，例如结合自己的业务等。您也曾提到，选择前端框架应综合考虑框架本身与团队情况。要考虑的点这么多，究竟怎样来综合考虑呢？具体的步骤应该是怎样的？

笔者：的确如此，技术本来是为业务服务的，单纯“玩”技术是没有前途，也找不到方向。前端框架之所以这么多，也是因为大家的业务侧重点各不相同。选择合适的框架，比选择一个时髦的框架重要多了。千万别让手下人员自行决定。他们玩不转可以“拍拍屁股走人”，留下一个烂摊子。我们要考虑到业务的可持续性、代码的可交接性及团队的普遍接受能力。比如一个公司，没有前端开发人员，都是后端开发人员顺手做前端的活，早期许多公司都是招 PHP 实现前端开发。他们的设计模式比较好，可以上手 angular。如果一个团队新人够多，不稳定，则只能用 jQuery 与 bootstrap。如果是一个创业公司，急着做出原型来拉投资，可以尝试 vue、avalon、react 等短平快的框架。但我所说的还是核心框架，涉及图表、UI 库，这些需要架构师见多识广，自己“趟过坑”，才让团队“集中过河”。

infoQ：有人说前端编程标准和方法渐渐出现稳定的趋势，您怎么看待这一观点？在之后的发展过程中，有没有可能标准完全统一？有没有可能某个前端框架“一统江湖”？

笔者：这个观点前半段是对的。像 jQuery 带来一系列便捷的操作 DOM 的方式，append、prepend、remove 等方法已经在 DOM4 中实现了。其最著名的选择器引擎，也有了原生替代品。因为浏览器商之间也存在竞争关系，将一些公认的好东西内置可以讨好用户。但浏览器厂商之间没怎么沟通，W3C 给出的规范也模糊，出现差异是在所难免的。因此不要相信浏览器，要使用框架！至于框架，框架之争是不会停息的，好的框架会不断涌现，它们可能以某个神奇的设计一下推翻前面的技术。就像 jQuery“灭掉”prototype，gulp“灭掉”grunt，webpack“灭掉”browserfy，react“灭掉”angular ……

并且没有一个框架覆盖所有需求，每个领域都有领头羊。因此想一个框架“一统江湖”，这个不可能也不实际。只要做好自己的擅长之处，开发就会比较顺利。

infoQ：您认为，前端开发人员学习框架设计应具备哪些能力？应从哪些方面着手进行设计？哪些地方有“坑”，需要注意避开？

笔者：这个问题比较笼统，我也只能笼统地回答。就像你问怎么挣大钱，有许多东西，人家说出来你也不能复制。首先，基础很重要，如计算机科班出来的人，搞前端就很容易上手。其次是设计模式，这是 Java 十多年积累的精华，是我们构建巨型工程的利器。现在前端框架的程序很多是上万行了。像过去那样，全是方法＋全局变量在堆砌，在生产环境中找 bug 是恶梦。最后是好好看高手们的框架，阅读源码是进步最快的方式之一。只有看了足够多的源码，你才能博采众长。再最后，就是宣传与测试了，宣传确保你拥有第一批用户，成为你继续维护与升级的动力。需要提供一系列便捷的下载渠道，因为酒香也怕巷子深。测试是确保你能留住用户。目前社区上有大量的测试工具，你可以将它们全部绑定在 webpack，在用户 build 工程时，把所有测试运行一遍。

最后说“坑”，其实没什么“坑”，所有浏览器兼容性问题与技术难点，许多技术高人都提供现成方案。但如果你做出框架，不再维护，对使用者来说这才是“大坑”。就是你不想维护了，就要找一个接盘者来主持。就像 jQuery、nodejs 等著名项目，原作者早早交给他人维护。

我今天的分享就到这里，谢谢大家！

欢迎来到异步社区！

异步社区的来历

异步社区（www.epubit.com.cn）是人民邮电出版社旗下 IT 专业图书旗舰社区，于 2015 年 8 月上线运营。

异步社区依托于人民邮电出版社 20 余年的 IT 专业优质出版资源和编辑策划团队，打造传统出版与电子出版和自出版结合、纸质书与电子书结合、传统印刷与 POD 按需印刷结合的出版平台，提供最新技术资讯，为作者和读者打造交流互动的平台。

社区里都有什么？

购买图书

我们出版的图书涵盖主流 IT 技术，在编程语言、Web 技术、数据科学等领域有众多经典畅销图书。社区现已上线图书 1000 余种，电子书 400 多种，部分新书实现纸书、电子书同步出版。我们还会定期发布新书书讯。

下载资源

社区内提供随书附赠的资源，如书中的案例或程序源代码。

另外，社区还提供了大量的免费电子书，只要注册成为社区用户就可以免费下载。

与作译者互动

很多图书的作译者已经入驻社区，您可以关注他们，咨询技术问题；可以阅读不断更新的技术文章，听作译者和编辑畅聊好书背后有趣的故事；还可以参与社区的作者访谈栏目，向您关注的作者提出采访题目。

灵活优惠的购书

您可以方便地下单购买纸质图书或电子图书，纸质图书直接从人民邮电出版社书库发货，电子书提供多种阅读格式。

对于重磅新书，社区提供预售和新书首发服务，用户可以第一时间买到心仪的新书。

用户账户中的积分可以用于购书优惠。100 积分 =1 元，购买图书时，在 0 使用积分 里填入可使用的积分数值，即可扣减相应金额。

特 别 优 惠

购买本书的读者专享异步社区购书优惠券。

使用方法：注册成为社区用户，在下单购书时输入 S4XC5 使用优惠码，然后点击“使用优惠码”，即可在原折扣基础上享受全单9折优惠。（订单满39元即可使用，本优惠券只可使用一次）

纸电图书组合购买

社区独家提供纸质图书和电子书组合购买方式，价格优惠，一次购买，多种阅读选择。

社区里还可以做什么？

提交勘误

您可以在图书页面下方提交勘误，每条勘误被确认后可以获得100积分。热心勘误的读者还有机会参与书稿的审校和翻译工作。

写作

社区提供基于Markdown的写作环境，喜欢写作的您可以在此一试身手，在社区里分享您的技术心得和读书体会，更可以体验自出版的乐趣，轻松实现出版的梦想。

如果成为社区认证作译者，还可以享受异步社区提供的作者专享特色服务。

会议活动早知道

您可以掌握IT圈的技术会议资讯，更有机会免费获赠大会门票。

加入异步

扫描任意二维码都能找到我们：

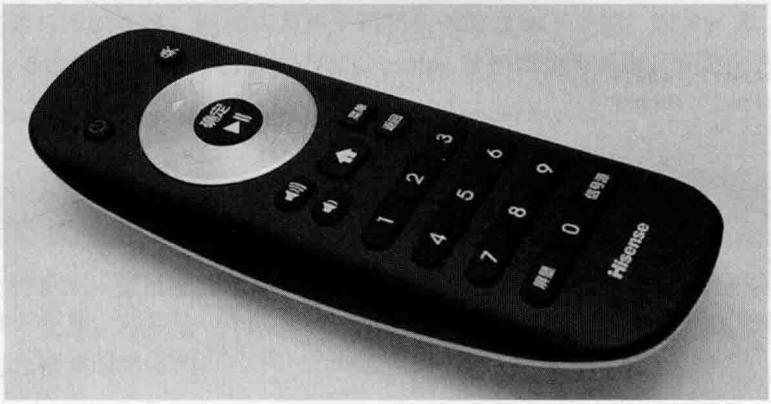
异步社区	微信服务号	微信订阅号	官方微博	QQ群：436746675

社区网址：www.epubit.com.cn

投稿 & 咨询：contact@epubit.com.cn

目　录

第1章 种子模块



1.1 模块化

最近几年，JavaScript 得到飞速发展，一些框架越来越大，已经不像过去那样全部写进一个 JS 文件中。但拆到多个 JS 文件时，就要决定哪个是入口文件，哪个是次要文件，而这些次要文件也不可能按 1、2、3、4 的顺序组织起来，可能 1 依赖于 2、3，3 依赖于 4、5，每个文件的顶部都像其他语言那样声明其依赖，最后在结束时说明如何暴露出那些变量或方法给外部使用。

就算你的框架只有几千行，在开发时将它们按功能拆分为 10 多个文件，维护起来也非常方便。加之 Node.js 的盛行，ES2016 许多语法不断被浏览器支持，我们更应该拥抱模块化。

本书所介绍的所有模块，都以 Node.js 提倡的 CommonJS 方式组织起来。

时下流行 3 种定义模块的规范：AMD[①]、CommonJS[②]与 ES6 module。它们都被 webpack 所支持。以 AMD 定义 JS 模块通过 RequireJS 能直接运行于浏览器；CommonJS 则需要 browserfy 等 Node.js 打包后才能运行于浏览器；ES6 module 在我写书时，还没有浏览器支持，需要 webpack、rollup 等

① AMD 全称是 Asynchronous Module Definition，即异步模块加载机制。从它的规范描述页面看，AMD 很短也很简单，但它却完整描述了模块的定义、依赖关系、引用关系以及加载机制。从它被 RequireJS、Node.js、dojo、jQuery 的使用中也可以看出它具有很大的价值。

② CommonJS 规范是 JavaScript 在服务器端的规范，Node.js 采用了这个规范。根据 CommonJS 规范，一个单独的文件就是一个模块。加载模块使用 require 方法。该方法读取一个文件并执行，最后返回文件内部的 exports 对象。CommonJS 构建的这套模块导出和引入机制使得用户完全不必考虑变量污染、命名空间等方案与之相比相形见绌。

Node.js 工具打包才能运行于浏览器。

下面简略演示一下这 3 种模块的定义方式。

```
//AMD
define(['./aaa', './bbb'], function(a, b){
   return {
       c: a + b
   }
})


// CommonJS
var a = require('./aaa')
var b = require('./bbb')
module.exports = {
    c: a + b
}


//es6 module
import a form './aaa'
import b form './bbb'
var c = a + b
export {c}
```

有关这 3 种模块的用法，后面的章节会详述，这里暂时略过。

本人是比较倾向使用 CommonJS 的，因为其相关的打包工具非常成熟，并且以这种方式编写的框架，可以与大量 Node.js 编写的测试框架无缝配合使用。

再说回来，刚才提到的入口文件。整个程序就是引入其他子模块，然后导出代表命名空间的 JavaScript 对象就行了。

```
var avalon = require('./seed/index')

require('./filters/index')
require('./vdom/index')
require('./dom/index')
require('./directives/index')
require('./strategy/index')
require('./component/index')
require('./vmodel/index')

module.exports = avalon
```

我们编写一个框架时，可能拆成上百个 JS 文件。为了方便寻找，这时需要按照功能或层次放进不同的子文件夹。然后每个子文件都有它的入口文件（index.js），由它来组织其依赖。本章的种子模块，就放到 seed 这个文件夹下了。

1.2 功能介绍

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