sitepoint

Broadview
www.broadview.com.cn 博文视点

JavaScript艺术与科学

The Art &Science of JavaScript

Cameron Adams
James Edwards
Christian Heilmann
Michael Mahemoff 著
Ara Pehlivanian
Dan Webb
Simon Willison
郑文涛 译

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

全世界最优秀技术人员给你带来的富有启发性的、最前沿的JavaScript

JavaScript艺术与科学

The Art &Science of JavaScript

Cameron Adams
James Edwards
Christian Heilmann
Michael Mahemoff 著
Ara Pehlivanian
Dan Webb
Simon Willison

郑文涛 译

電子工業出版社
Publishing House of Electronics Industry
北京 · BEIJING

内 容 简 介

本书由七位 JavaScript 领域的专家合作撰写，图文并茂，理论与实践紧密结合，通过大量示例代码帮助读者更好地理解和掌握最前沿的 JavaScript 知识与技巧。书中内容一共分为七章，涵盖表格处理、客户端 Badge、canvas 绘图、Firebug、元编程、3D 迷宫以及混搭，囊括了近些年 JavaScript 用于网页应用开发的最佳实践与实用技巧。本书作为 JavaScript 中级读物，主要面向具有一定 JavaScript 经验的网页开发者，旨在帮助他们将 JavaScript 水平提升到一个新的高度。

版权贸易合同登记号　图字：01-2009-1486

图书在版编目（CIP）数据

JavaScript 艺术与科学 / 亚当斯（Adams,C.）等著；郑文涛译. —北京：电子工业出版社，2010.10
书名原文：The Art & Science of JavaScript
ISBN 978-7-121-11936-1
Ⅰ. ①J… Ⅱ. ①亚… ②郑… Ⅲ. ①JAVA 语言－程序设计 Ⅳ. ①TP312
中国版本图书馆 CIP 数据核字(2010)第 191216 号

策划编辑：卢鹞翔
责任编辑：杨绣国
印　　刷：北京天宇星印刷厂
装　　订：三河市皇庄路通装订厂
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　　**邮编**：100036
开　　本：787×1092　1/16　**印张**：17.5　**字数**：409 千字
印　　次：2010 年 10 月第 1 次印刷
定　　价：49.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。
质量投诉请发邮件至 *zlts@phei.com.cn*，盗版侵权举报请发邮件至 *dbqq@phei.com.cn*。服务热线：（010）88258888。

前言

Preface

曾几何时，JavaScript 是个不光彩的词。

在网页刚诞生的早期，开发者只是用 JavaScript 生成一些无聊的动画或是华而不实的玩意儿。这些不当甚至是错误的用法给它带来了不好的名声。

幸好，那些日子已经远离我们，而本书将向你展示的是最近的发展。其中所折射的是 JavaScript 发展过程中的转折点——许多涉及到的效果和技巧在几年前还被认为是不可能实现的。

作为一门逐渐成熟起来的编程语言，JavaScript 已经非常流行，随之出现了许多非常实用的技巧，并基于此演化出大量软件框架。作为长期的 JavaScript 爱好者，我们一直认为它具有很大的潜力。如今，优秀网页应用的点睛之处往往是用 JavaScript 实现的。如果说 CSS 是二十一世纪头几年的宠儿，那么 JavaScript 已经毫无争议的取代了 CSS 的地位。

在本书中，我们组成了一个专家团队——他们都是 JavaScript 开发者中名副其实的专家——帮你将 JavaScript 技能提升到更高的水平。你即将打开潘多拉的魔盒，展现在你面前的包括让人印象深刻的 mashup、令人惊叹的动态图形，以及细致入微的用户体验提升。一旦你了解到 JavaScript 所具备的能力，便可以用书中的代码去为用户创造惊奇的用户体验。当然，如果有心，你也可以用学到的知识改变整个世界。

我们期待到你的网站发布派对上喝一杯！

适用读者

Who Should Read This Book?

本书面向中等水平的 JavaScript 开发人员，在不牺牲网页可用性和实用性的前提下，帮助他们提高 JavaScript 技能水平。如果以前从没写过一行 JavaScript 代码，这本书可能不太适合你，后面章节中的某些逻辑对你来说会有点难。

如果你只有少量的 JavaScript 经验，却非常熟悉另外一门编程语言（比如 PHP 或 Java），那也没问题——我们会手把手地教，你可以下载所有的代码并亲自实践。如果你已经是一个很有经验的 JavaScript 开发人员，如果你一无所获，我们会非常非常吃惊。坦白地讲，如果仅仅学到一点点，你应当到 SitePoint 联系我们——可能会有一个书籍项目等你搞定。

内容组织

What's Covered in This Book?

第 1 章：玩转表格

HTML 表格在网页开发者中的名声很坏，这可能是因为多年来人们将其错误地使用在页面布局中，亦或是因为它们用起来太无聊了。在这一章，Ara Pehlivanian 会证明，合理地使用表格不仅不会无聊，反而会带来很多乐趣，尤其是结合了 JavaScript 以后。Ara 首先会介绍 DOM，然后展示如何制作用鼠标或键盘进行排序和拖拽的表格列。

第 2 章：创建客户端 Badge

Badge 是指那些可以放到你博客上以增加个性化的第三方数据（比如缩略图、链接等）片段。Christian Heilmann 将带我们经历从无到有使用 JSON 制作 Badge 的整个过程，如果第三方服务器的连接失败，还有第二套应急方案。

第 3 章：用 canvas 绘制向量图形

在这一章，Cameron Adams 会介绍 canvas 元素，并展示如何用它构建所有现代浏览器都支持的向量图形：从静态图示，到数据库驱动的结构图和饼图。读完之后，你再也不会用老眼光看待网页里的图形。

第 4 章：用 Firebug 进行调试和测定

Firebug 是 Firefox 浏览器的一个插件，但是仅把它叫做插件是不公平的——Firebug 是一个全功能的编辑、调试、测定工具。它将曾经痛苦的 JavaScript 调试优化工作变得更加简单、有趣。这里，Michael Mahemoff 将揭示大量专家级的技巧和隐藏的知识，让你更加深入地了解这个不可缺少的开发工具。

第 5 章：用 JavaScript 进行元编程

这里，Dan Webb 带我们走入 JavaScript 语言本身的原理之中。通过了解一些元编程的理论，他将展示如何使用 JavaScript 扩展语言自身，提升面向对象的能力，增强对老版本浏览器的支持，并增加一些方法和操作符让 JavaScript 开发更方便。

第 6 章：用 CSS 和 JavaScript 构建三维迷宫

就当你觉得已经了解了全部的时候，James Edwards 通过制作一个实际的游戏——供玩家探索三维迷宫！——向你展示了如何将 CSS 和 JavaScript 技术发挥到极致。这章结尾介绍了一个平面图生成器和一些提高可用性的功能，比如键盘导航和说明。通过这些内容，本章强调了这样一个事实：JavaScript 的潜力很大，没有做不到，只有想不到。

第 7 章：Flickr 和 Google 地图的混搭

你有想过结合最好的照片管理网站 Flickr 和最好的地图服务网站 Google 地图，来制作一个属于自己的高级应用么？你可以的。Simon Willison 会向你展示：借助于 JavaScript API，结合两个第三方网站制作一个 mashup 比你想象的容易。

配套网站

The Book's Web Site

本书的配套网站在 http://www.sitepoint.com/books/jsdesign1/，你可以通过它获取如下资源。

代码存档

The Code Archive

在阅读本书的过程中，你会注意到很多代码样本上面有文件名。它们对应于代码库（一个包含本书中所有完整实例的 ZIP 下载包）中的相关文件。你可以点击配套网站上的 Code Archive 链接下载。

更新和勘误

Updates and Errata

错误是难免的，细心的读者一定会在本书中发现至少一到两处错误。配套网站上的 Corrections and Typos 页面提供了最新的排版和代码错误，并会根据最新发布的浏览器和相关标准提供必要的更新[1]。

SitePoint 论坛

The SitePoint Forums

如果你想和其他网页开发者交流关于本书的想法，你应当加入 SitePoint 的在线社区[2]。特别是 JavaScript 论坛[3]，提供了丰富的超越本书解决方案的信息，还有很多有趣并且经验丰富的 JavaScript 开发者。在这里，你能学到新的技巧，快速获得解答，并得到快乐。

SitePoint 新闻邮件

The SitePoint Newsletters

除了出版书籍，SitePoint 还会发布免费的电子邮件新闻，包括 The SitePoint Tribune、The Site-Point Tech Times 和 The SitePoint Design View。它们可以让你获取最新的关于网页开发的新闻、产品发布、趋势、提示和技巧。赶紧去 http://www.sitepoint.com/newsletter/注册吧。

---

1 http://www.sitepoint.com/books/jsdesign1/errata.php

2 http://www.sitepoint.com/forums/

3 http://www.sitepoint.com/launch/jsforum/

反馈

Your Feedback

如果论坛不能帮你解决问题，或者你因为其他事情想联系我们，最好的方式是给books@sitepoint.com 发送电子邮件。我们有一个电子邮件支持系统跟踪你的请求，以及热情的工作人员可以解答你的问题。我们特别欢迎改进建议或是你发现的错误。

规范

Convertions Used in This Book

为了标记不同类型的信息，我们在本书中使用了一些排版和布局风格。注意如下几点。

代码样例

Code Samples

本书中的代码用等宽字体显示，如下：

```
<h1>A perfect summer's day</h1><p>It
    was a lovely day for a walk in the park. The birds were
    singing and the kids were all back at school.</p>
```

如果代码被包含在代码存档里，代码上方会显示对应的文件名，如下：

example.css

```
.footer { background-color: #CCC; border-top: 1px
        solid #333; }
```

如果只显示了文件中的一部分代码，文件名后面会加上 excerpt：

example.css (*excerpt*)

```
border-top: 1px solid #333;
```

有些代码应当写成一行，但是由于页面空间的限制，我们需要做断行处理。→用来表示这样的换行，它实际代码中是不存在的。垂直省略号（⋮）表示为了节省空间而省略掉的代码。

```
if (a == b) {
 ⋮
}
URL.open.("http://www.sitepoint.com/blogs/2007/11/01/the-php-anthology-101-essent
  →ial-tips-tricks-hacks-2nd-edition");
```

提示、记录和警告

Tips, Notes, and Warnings

嘿，你！

对你有帮助的小提示。

呃，打扰一下

有用的信息，它们和主题相关，但并不关键。你可以把它们当作额外的补充信息。

你要确保……

……注意这些重要的信息。

当心！

这些警告会着重强调你可能会遇到的陷阱。

目录

Table of Contents

前言 I

适用读者 I

内容组织 II

配套网站 III

代码存档 III

更新和勘误 III

SitePoint 论坛 III

SitePoint 新闻邮件 III

反馈 IV

规范 IV

代码样例 IV

提示、记录和警告 V

第 1 章　玩转表格 1

1.1　表格剖析 1

1.1.1　用 getElementById 访问表格元素 4

1.1.2　用 getElementsByTagName 访问表格元素 6

1.2　按列排序 7

1.2.1　让表格可排序 7

1.2.2　执行排序 12

1.3　创建可拖拽的列 24

1.3.1　让表格列可拖拽 25

1.3.2　不用鼠标也能拖拽表格列 37

1.4　小结 44

第 2 章　创建客户端 Badge 45

2.1　Badge 简介 46

2.1.1　Badge 太多会坏事 46

2.1.2　现成的 Badge 48

2.1.3　服务器端 badge 50

2.1.4 自定义客户端 Badge 51
2.2 客户端 Badge 的选择：Ajax 和 JSON 53
2.2.1 Ajax 的问题 53
2.2.2 JSON：轻量的原生数据格式 54
2.2.3 提供连接失败的处理 58
2.3 规划 badge 脚本 59
2.4 完整的 badge 脚本 61
2.4.1 定义配置变量 63
2.4.2 定义公共方法 64
2.4.3 定义私有方法 67
2.5 请求服务器备份 72
2.6 小结 73

第 3 章 用 canvas 绘制向量图形 75
3.1 使用 canvas 76
3.1.1 canvas API 77
3.1.2 了解向量图形 78
3.1.3 创建形状 79
3.2 创建饼图 98
3.2.1 绘制饼图 98
3.2.2 投射阴影 104
3.2.3 动态更新饼图 109
3.3 Internet Explorer 中的 canvas 115
3.4 小结 119

第 4 章 用 Firebug 进行调试和测定 121
4.1 安装并运行 Firebug 122
4.1.1 安装 Firefox 和 Firebug 122
4.1.2 Firebug 初体验 123
4.1.3 打开、关闭 Firebug，并调整其大小 124
4.1.4 启用和禁用 Firebug 127
4.2 Firebug 组成部分 127
4.2.1 公共组件 127

4.2.2 Firebug 视图 128
4.2.3 切换视图 132
4.3 使用 Firebug 133
4.3.1 执行快速应用开发 133
4.3.2 利用 Console 进行监控、记录日志和运行 134
4.3.3 实时查看和编辑 138
4.3.4 调试你的应用 140
4.3.5 优化你的应用性能 143
4.4 相关工具 145
4.4.1 Firebug Lite 145
4.4.2 YSlow 146
4.4.3 微软的工具 146
4.4.4 其他 Firefox 扩展 147
4.5 小结 147

第 5 章 用 JavaScript 进行元编程 149
5.1 基石 150
5.1.1 （几乎）所有东西都是哈希 150
5.1.2 在对象里查找和遍历属性 151
5.1.3 检测类型 152
5.1.4 JavaScript 里面没有类 153
5.1.5 检测一个函数是否以 new 调用 154
5.1.6 函数就是对象 155
5.1.7 理解 arguments 数组 157
5.1.8 理解闭包 159
5.2 元编程技巧 164
5.2.1 创建带有默认参数的函数 164
5.2.2 处理内建函数及变量 165
5.2.3 创建自优化的函数 168
5.2.4 在鞋带上使用面向特征编程 171
5.2.5 用动态函数构建更好的 API 172
5.2.6 创建动态的构造器 176
5.2.7 模拟传统的面向对象 178

5.3 小结 187

第 6 章 用 CSS 和 JavaScript 构建三维迷宫 189

6.1 基本原理 190

6.1.1 制作三角形 191

6.1.2 定义地面设计 193

6.1.3 创建透视效果 196

6.2 制作动态视图 198

6.2.1 核心方法 198

6.2.2 使用最后几手 208

6.2.3 该方法的局限性 209

6.3 创建地图视图 209

6.4 添加说明 212

6.5 地面设计 213

6.6 进一步开发 214

6.6.1 使用回调 214

6.6.2 无限的可能性 215

6.7 小结 216

第 7 章 Flickr 和 Google 地图的混搭 217

7.1 API、混搭和小控件！天哪！ 218

7.1.1 Flickr 和 Google 地图 218

7.1.2 绘制地图 219

7.1.3 打上地理标签的照片 221

7.1.4 获取数据 222

7.2 JSON 223

7.2.1 同源限制 224

7.3 把它们放在一起 233

7.3.1 增强我们的小部件 238

7.3.2 把所有的放在一起 245

7.4 下一步 249

7.5 小结 250

索引 251

第1章

玩转表格

Fun with Tables

在很长一段时间，当网页设计师须要以非线性的方式来布局网页内容时，会选择表格作为工具。虽然表格不应该被这样使用，但是这种行-列结构提供了一种自然的网格系统，对设计师来说诱惑难当。这种对表格的误用，转移了许多设计师的注意力，他们不再关注表格最初被设计的用途：标记表格数据。

虽然表格最初诞生于 HTML，但它并不局限于此。有了 JavaScript，我们可以为原本静态的 HTML 表格增加交互功能。本章的目标是让你充分理解如何使用 JavaScript 操作表格，一旦你掌握了基本的知识，便能够从容地处理那些狂野的、超越本章示例的情况。

如果你是个 DOM 新手，会发现本章也介绍了 DOM 的操作技巧，这些我都会尽可能解释清楚。

1.1 表格剖析

Anatomy of a Table

在体验表格的乐趣之前，我们先要掌握一些基本知识。在我们对表格的结构有了很好的理解之后，才能用 JavaScript 对它进行简单、高效的操作。

在开头我提到了表格的行-列结构。事实上，表格并没有列这样的实体——至少在 HTML 表

格里没有。列只是一个假象。从结构上看，一个表格是一组行的集合，其中每行又是一组格子的集合。没有一个实实在在的 HTML 元素用来表示一列格子——最接近的元素只是 colgroup 和 col，但是它们的存在仅仅是为了更好地描述表格样式。从实际结构上来说，是没有列的。

让我们仔细看看如图 1.1 所示的简单表格。

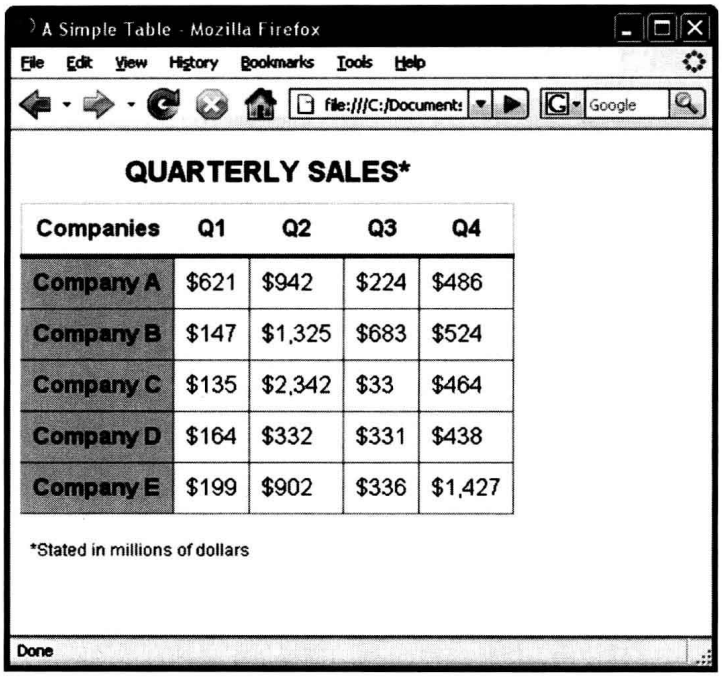

图 1.1　一个简单表格

为了让这个表格看上去更舒服一些，我给它增加了一些 CSS 样式。标记如下所示：

simple.html (*excerpt*)

```
<table id="sales" summary="Quarterly Sales figures for competing companies.
The figures are stated in millions of dollars.">
  <caption>Quarterly Sales*</caption>
  <thead>
    <tr>
      <th scope="col">Companies</th>
      <th scope="col">Q1</th>
      <th scope="col">Q2</th>
      <th scope="col">Q3</th>
      <th scope="col">Q4</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <th>Company A</th>
      <td>$621</td>
```

```
      <td>$942</td>
      <td>$224</td>
      <td>$486</td>
    </tr>
    <tr>
      <th scope="row">Company B</th>
      <td>$147</td>
      <td>$1,325</td>
      <td>$683</td>
      <td>$524</td>
    </tr>
    <tr>
      <th scope="row">Company C</th>
      <td>$135</td>
      <td>$2,342</td>
      <td>$33</td>
      <td>$464</td>
    </tr>
    <tr>
      <th scope="row">Company D</th>
      <td>$164</td>
      <td>$332</td>
      <td>$331</td>
      <td>$438</td>
    </tr>
    <tr>
      <th scope="row">Company E</th>
      <td>$199</td>
      <td>$902</td>
      <td>$336</td>
      <td>$1,427</td>
    </tr>
  </tbody>
</table>
<p class="footnote">*Stated in millions of dollars</p>
```

每组<tr></tr>标签告诉浏览器在表格中开始新的一行。其中<th>和<td>标签表示标题格和数据格。虽然这些格子在 HTML 里是垂直排列的，很像数据列，但实际它们是作为行的一部分以水平方式排列的。

同时，我们注意到这些行通过<thead>或<tbody>组织在一起。这样做不仅提供了更清晰的语义结构，还可以让 JavaScript 操作表格更加容易，稍后我们会讲到这点。

1.1.1　用 getElementById 访问表格元素

Accessing Table Elements with getElementById

当浏览器渲染页面的时候，会为其创建一棵 DOM 树。有了这棵 DOM 树，我们便能够使用一些原生的 DOM 方法访问表格里的元素。

`getElementById` 是本章中使用最多的方法。下面的例子展示了如何用它访问表格的一行：

```
var sales = document.getElementById("sales");
var rows = sales.rows;
```

在上述代码中，我们先用 DOM 方法 `getElementById` 获取表格的引用，存放于变量 `sales`。然后用该变量获取表格中所有的行，存放于变量 `rows`。

这个例子没什么问题，不过如果我们只想获取元素 `thead` 或 `tbody` 中的行呢？其实，这两个行的集合在 DOM 树中也有所反映，我们可以通过如下代码访问它们：

```
var sales = document.getElementById("sales");
var headRow = sales.tHead.rows;
var bodyRows = sales.tBodies[0].rows;
```

如其所示，访问 `thead` 中的行集合直接明了。不过，访问 `tbody` 中的行集合稍有不同，因为一个表格可能包含多个 `tbody`。我们在这里获取的是 `tBodies` 中第一个 `tbody` 所包含的行集合。由于 `tBodies` 中的元素个数从 0 开始，就像数组一样，所以其中的第一项就是 `0` 项，可以通过 `tBodies[0]`获得。

什么是 DOM？

Document Object Model (DOM)是一个标准化的用于操作如 HTML 和 XML 标记语言的编程接口。

简单地说，DOM 是文档的面向对象表示。DOM 树中的一个对象代表了 HTML 文档中的一个元素。这些对象——通常叫做节点——按照其对应的 HTML 元素及嵌套关系组织在一起，就像一棵树。

DOM 树还包含一些其他对象，以方便我们访问和操作文档：比如示例代码中的 rows 对象，在源 HTML 文档中并不存在。DOM 中的每个对象还包含一些补充信息，比如位置、内容及物理尺寸。

按照同样的原理可以访问特定的一行。让我们从第一个 tbody 中获取第一行：

```
var sales = document.getElementById("sales");
var bodyRows = sales.tBodies[0].rows;
var row = bodyRows[0];
```

当然，条条大路通罗马。看看下面这个例子：

```
var sales = document.getElementById("sales");
var tBody = sales.tBodies[0];
var rows = tBody.rows;
var row = rows[0];
```

下面的一行代码也可以得到同样的结果：

```
var row = document.getElementById("sales").tBodies[0].rows[0];
```

当你选择具体的方法时应当兼顾效率和可读性。用四行代码访问表格的某一行可能太冗长了，但是只写一行读起来却很费力。上面例子中的单行代码的容错能力不如四行代码，因为我们没有办法在访问某个对象的子节点之前检查该对象是否存在。

当然，你可以走向另一个极端：

```
var sales = document.getElementById("sales");
if (sales) {
  var tBody = sales.tBodies[0];
  if (tBody) {
    var rows = tBody.rows;
    if (rows) {
      var row = rows[0];
    }
  }
}
```

这段代码在执行每一步之前都检查结果的正确性，使它成为上述四个示例中最鲁棒的。毕竟，你所访问的表格可能不包含 tbody 元素，或者连一行都没有。通常，在鲁棒和简洁之间，我更重视鲁棒。如果像之前的单行代码那样，不检查 tbody 是否存在就直接获取它的第一行，在某些用户那里可能会导致未捕获的错误。在后面几节中，我们会探讨一些准则，用于选择合适的编码策略。