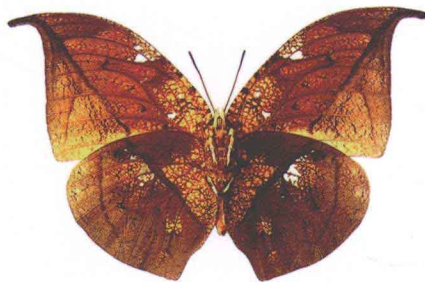


日本亚马逊2012年度销量No.1



JavaScript 编程全解

对应 ECMAScript 最新版本

完美涵盖AJAX / jQuery

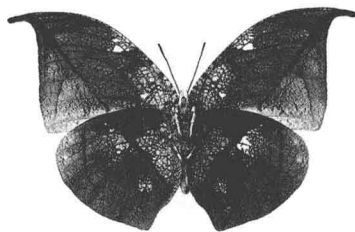
/ Web API / Node.js

【日】井上诚一郎 土江拓郎 滨边将太 著
陈筱烟 译



人民邮电出版社
POSTS & TELECOM PRESS

TURING 图灵程序设计丛书



JavaScript

编程全解

【日】井上诚一郎 土江拓郎 滨边将太 著
陈筱烟 译

人民邮电出版社
北 京

图书在版编目(CIP)数据

JavaScript 编程全解 / (日) 井上诚一郎, (日) 土江拓郎, (日) 滨边将太著; 陈筱烟译. -- 北京: 人民邮电出版社, 2013.12

(图灵程序设计丛书)

ISBN 978-7-115-33341-4

I. ①J… II. ①井… ②土… ③滨… ④陈… III. ①

JAVA 语言—程序设计 IV. ①TP312

中国版本图书馆CIP数据核字(2013)第240740号

内 容 提 要

本书涵盖了JavaScript开发中各个方面的主题,对从客户端及服务器端JavaScript等基础内容,到HTML5、Web API、Node.js与WebSocket等热门技术,都作了深入浅出的介绍与说明。读者能够通过本书了解当今JavaScript开发的最新现状。本书的一大特色是对JavaScript语言的语法规则进行了细致的说明,并通过大量纯正的JavaScript风格代码,帮助读者准确地掌握JavaScript的语言特性及细节用法。

本书适合JavaScript开发初学者系统入门、有经验的JavaScript开发者深入理解语言本质,也适合开发团队负责人、项目负责人作为综合性的JavaScript参考书阅读。

◆ 著 [日]井上诚一郎 土江拓郎 滨边将太
译 陈筱烟
责任编辑 乐 馨
执行编辑 徐 骞
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>
北京鑫正大印刷有限公司印刷

◆ 开本: 800×1000 1/16
印张: 26.25
字数: 794千字 2013年12月第1版
印数: 1-3 000册 2013年12月北京第1次印刷
著作权合同登记号 图字: 01-2013-3125号

定价: 79.00元

读者服务热线: (010)51095186转600 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

版 权 声 明

PERFECT JavaScript by Seiichiro Inoue, Takuro Tsuchie, Shota Hamabe

Copyright 2011 Seiichiro Inoue, Takuro Tsuchie, Shota Hamabe

All rights reserved.

Original Japanese edition published by Gijyutsu-Hyoron Co., Ltd., Tokyo

This Simplified Chinese language edition published by arrangement with

Gijyutsu-Hyoron Co., Ltd., Tokyo in care of Tuttle-Mori Agency, Inc., Tokyo

本书中文简体字版由 Gijyutsu-Hyoron Co., Ltd. 授权人民邮电出版社独家出版。未经出版者书面许可，不得以任何方式复制或抄袭本书内容。

版权所有，侵权必究。

注意

购买、使用本书前的必读事项

- 本书所记述的内容，仅仅是作者向读者提供的信息，因此，读者对本书内容的使用，基于其自身的判断，一切责任自负。对于使用本书内容所造成的结果，技术评论社、原书作者、人民邮电出版社以及译者，概不负责。
- 本书内容依据 2011 年 8 月 30 日的情况所写，因此在阅读本书时，个别内容可能已经发生了更改。对于部分变更，已经附上了译者注，帮助读者理解。关于书中出现的软件信息，如无特别说明，均以 2011 年 8 月 30 日的最新版为准。在软件版本升级之后，其功能与界面可能会与本书中的说明有所差异。在购买本书前，请务必确认软件版本号。
- 本书内容及其收录的范例代码，已确认能够在以下环境中运行。

操作系统	Windows 7 Professional 64位版
浏览器	Internet Explorer / Firefox / Google Chrome

在除此之外的环境中使用时，操作方法、软件界面以及程序的执行方式可能会与本书的记述有所不同，敬请谅解。

请在理解以上这些注意事项的基础上使用本书。

- 可以在下面的站点中找到本书的支持信息(日文)。

<http://gihyo.jp/book/2011/978-4-7741-4813-7/support>

※Microsoft及 Windows是美国微软公司在美国及其他国家的商标或注册商标。

※本书中介绍的商品名称，都是各相关公司的商标或注册商标。

前言

首先感谢您购买本书。

这是一本关于 JavaScript 程序设计语言的书。本书的前半部分将对 JavaScript 的语言基础进行解说，而后半部分主要介绍包括客户端 JavaScript、HTML5、Web API 以及服务器 JavaScript 等与 JavaScript 相关的应用领域。

本书面向有一定编程基础的开发者，因此书中的 JavaScript 代码都只取了片段。希望通过复制粘贴来使用这些代码的读者，在阅读本书时或许会感到有些困难。此外，本书中没有涉及网页设计和用户体验的内容，如果只是希望学习和网页显示效果有关的 JavaScript 知识，阅读本书并不合适。

本书的目标读者是希望深入学习 JavaScript 并开发完整的 Web 应用程序的人。那些平时主要使用 Java 或是 PHP 等其他语言的开发者，有时也会遇到一些不得不使用 JavaScript 语言的情况。对于这些开发者来说，如果你抱有“既然要使用 JavaScript，就应该学习正确的语言规范以写出良好代码”的想法，也推荐你阅读本书。

如果希望理解 JavaScript 的语言基础，请阅读本书的第 2 部分。为了让没有接触过程序设计语言的人也能理解，本书着实下了一番功夫。虽然其中包含了不少和 Java 对比的内容，不过只要按顺序认真阅读，初学者也不会感到吃力。

从第 3 部分开始本书将介绍 JavaScript 的应用，其中还包括 HTML5 和 Node.js 等热门的新技术。其实，如果真要理性地评价 JavaScript 这门程序设计语言，我们会发现它并不具有令人兴奋的特性。虽然 JavaScript 在其看似平凡的外表之下有着复杂的内部构造，但它确实不是一种采用了计算机科学领域最新技术的新型语言。事实上，学习 JavaScript 的意义与其说是为了学习这门语言本身，倒不如说是为了学习和使用其相关领域的知识。近年来，JavaScript 相关领域的发展着实令人着迷。毫不夸张地说，现在互联网的新热点大多都和 JavaScript 有关。如果本书的后半部分内容能够将这种兴奋感传达给各位读者的话，我将感到不胜荣幸。

井上诚一郎

2011 年 8 月 31 日

■目标读者

- 读过一些 JavaScript 的入门书籍，希望进一步了解 JavaScript 本质的人。
- 平时虽然常常使用 JavaScript，但还没有完全理解 JavaScript，并因此感到有些信心不足的人。
- 虽然主要使用其他的程序设计语言，但有时也会使用 JavaScript 的人。
- 认为 JavaScript 会是今后的主流语言的人。

目录 C O N T E N T S

第 1 部分 JavaScript 概要

001

第1章	JavaScript概要	002
1.1	JavaScript概要	002
1.2	JavaScript的历史	002
1.3	ECMAScript	003
1.3.1	JavaScript的标准化	003
1.3.2	被放弃的ECMAScript第4版	004
1.4	JavaScript的版本	004
1.5	JavaScript实现方式	005
1.6	JavaScript运行环境	006
1.6.1	核心语言	006
1.6.2	宿主对象	007
1.7	JavaScript相关环境	007
1.7.1	库	007
1.7.2	源代码压缩	007
1.7.3	集成开发环境	008

第 2 部分 JavaScript 的语言基础

009

第2章	JavaScript基础	010
2.1	JavaScript的特点	010
2.2	关于编排格式	011
2.3	变量的基础	012
2.3.1	变量的使用方法	012
2.3.2	省略var	013
2.3.3	常量	013
2.4	函数基础	013
2.4.1	函数的定义	013
2.4.2	函数的声明与调用	014
2.4.3	匿名函数	014
2.4.4	函数是一种对象	015
2.5	对象的基础	016
2.5.1	对象的定义	016
2.5.2	对象字面量表达式与对象的使用	016
2.5.3	属性访问	017
2.5.4	属性访问（括号方式）	017
2.5.5	方法	018
2.5.6	new表达式	018
2.5.7	类与实例	018
2.5.8	对类的功能的整理	018
2.5.9	对象与类型	019
2.6	数组的基础	019
第3章	JavaScript的数据类型	021
3.1	数据类型的定义	021
3.1.1	在数据类型方面与Java作比较	021
3.1.2	基本数据类型和引用类型	022
3.2	内建数据类型概要	022
3.3	字符串型	023
3.3.1	字符串字面量	023
3.3.2	字符串型的运算	024
3.3.3	字符串型的比较	024
3.3.4	字符串类（String类）	025
3.3.5	字符串对象	026
3.3.6	避免混用字符串值和字符串对象	027
3.3.7	调用String函数	027
3.3.8	String类的功能	027
3.3.9	非破坏性的方法	029
3.4	数值型	029
3.4.1	数值字面量	029
3.4.2	数值型的运算	030
3.4.3	有关浮点数的常见注意事项	030

3.4.4	数值类 (Number类)	031
3.4.5	调用Number函数	031
3.4.6	Number类的功能	032
3.4.7	边界值与特殊数值	033
3.4.8	NaN	034
3.5	布尔型	035
3.5.1	布尔值	035
3.5.2	布尔类 (Boolean类)	036
3.5.3	Boolean类的功能	036
3.6	null型	037
3.7	undefined型	037
3.8	Object类型	038
3.9	数据类型转换	039
3.9.1	从字符串值转换为数值	039
3.9.2	从数值转换为字符串值	040
3.9.3	数据类型转换的惯用方法	040
3.9.4	转换为布尔型	041
3.9.5	其他的数据类型转换	042
3.9.6	从Object类型转换为基本数据类型	042
3.9.7	从基本数据类型转换为Object类型	043

第4章

语句、表达式和运算符

045

4.1	表达式和语句的构成	045
4.2	保留字	045
4.3	标识符	046
4.4	字面量	047
4.5	语句	047
4.6	代码块 (复合语句)	048
4.7	变量声明语句	048
4.8	函数声明语句	048
4.9	表达式语句	048
4.10	空语句	049
4.11	控制语句	049
4.12	if-else语句	050
4.13	switch-case语句	052
4.14	循环语句	054
4.15	while语句	055
4.16	do-while语句	056
4.17	for语句	057
4.18	for in语句	058
4.18.1	数列与for in语句	059
4.18.2	在使用for in语句时需要注意的地方	060
4.19	for each in 语句	060
4.20	break语句	061
4.21	continue语句	061
4.22	通过标签跳转	062
4.23	return语句	063
4.24	异常	063
4.25	其他	064
4.26	注释	065
4.27	表达式	065
4.28	运算符	065
4.29	表达式求值	066
4.30	运算符的优先级以及结合律	066
4.31	算术运算符	067
4.32	字符串连接运算符	068
4.33	相等运算符	068
4.34	比较运算符	069
4.35	in运算符	070
4.36	instanceof运算符	071
4.37	逻辑运算符	071
4.38	位运算符	072
4.39	赋值运算符	072
4.40	算术赋值运算符	073
4.41	条件运算符 (三目运算符)	073
4.42	typeof运算符	073
4.43	new运算符	074
4.44	delete运算符	074
4.45	void运算符	074
4.46	逗号 (,) 运算符	074
4.47	点运算符和中括号运算符	075

4.48	函数调用运算符	075
4.49	运算符使用以及数据类型转换中需要注意的地方	075

第5章 变量与对象 076

5.1	变量的声明	076
5.2	变量与引用	076
5.2.1	函数的参数（值的传递）	078
5.2.2	字符串与引用	079
5.2.3	对象与引用相关的术语总结	079
5.3	变量与属性	080
5.4	变量的查找	081
5.5	对变量是否存在的检验	081
5.6	对象的定义	082
5.6.1	抽象数据类型与面向对象	082
5.6.2	实例间的协作关系与面向对象	083
5.6.3	JavaScript的对象	083
5.7	对象的生成	083
5.7.1	对象字面量	083
5.7.2	构造函数与new表达式	085
5.7.3	构造函数与类的定义	087
5.8	属性的访问	087
5.8.1	属性值的更新	088
5.8.2	点运算符与中括号运算符在使用上的区别	088
5.8.3	属性的枚举	089
5.9	作为关联数组的对象	089
5.9.1	关联数组	089
5.9.2	作为关联数组的对象的注意点	090
5.10	属性的属性	091
5.11	垃圾回收	092
5.12	不可变对象	092
5.12.1	不可变对象的定义	092
5.12.2	不可变对象的作用	092
5.12.3	实现不可变对象的方式	093
5.13	方法	094
5.14	this引用	094
5.14.1	this引用的规则	094
5.14.2	this引用的注意点	095
5.15	apply与call	096
5.16	原型继承	097
5.16.1	原型链	097
5.16.2	原型链的具体示例	099
5.16.3	原型继承与类	100
5.16.4	对于原型链的常见误解以及_proto_属性	100
5.16.5	原型对象	101
5.16.6	ECMAScript第5版与原型对象	101
5.17	对象与数据类型	102
5.17.1	数据类型判定（constructor属性）	102
5.17.2	constructor属性的注意点	102
5.17.3	数据类型判定（instance运算与isPrototypeOf方法）	103
5.17.4	数据类型判定（鸭子类型）	103
5.17.5	属性的枚举（原型继承的相关问题）	104
5.18	ECMAScript第5版中的Object类	105
5.18.1	属性对象	105
5.18.2	访问器的属性	106
5.19	标准对象	108
5.20	Object类	108
5.21	全局对象	110
5.21.1	全局对象与全局变量	110
5.21.2	Math对象	111
5.21.3	Error对象	112

第6章 函数与闭包 113

6.1	函数声明语句与匿名函数表达式	113
6.2	函数调用的分类	113
6.3	参数与局部变量	114
6.3.1	arguments对象	114
6.3.2	递归函数	114
6.4	作用域	115
6.4.1	浏览器与作用域	116
6.4.2	块级作用域	116
6.4.3	let与块级作用域	117

6.4.4	嵌套函数与作用域	119
6.4.5	变量隐藏	119
6.5	函数是一种对象	120
6.6	Function类	122
6.7	嵌套函数声明与闭包	123
6.7.1	对闭包的初步认识	123
6.7.2	闭包的原理	123
6.7.3	闭包中需要注意的地方	126
6.7.4	防范命名空间的污染	127
6.7.5	闭包与类	129
6.8	回调函数设计模式	130
6.8.1	回调函数与控制反转	130
6.8.2	JavaScript与回调函数	131

第7章**数据处理****134**

7.1	数组	134
7.1.1	JavaScript的数组	134
7.1.2	数组元素的访问	135
7.1.3	数组的长度	136
7.1.4	数组元素的枚举	136
7.1.5	多维数组	137
7.1.6	数组是一种对象	138
7.1.7	Array类	139
7.1.8	数组对象的意义	140
7.1.9	数组的习惯用法	141
7.1.10	数组的内部实现	144
7.1.11	数组风格的对象	145
7.1.12	迭代器	145
7.1.13	生成器	147
7.1.14	数组的内包	149
7.2	JSON	149
7.2.1	JSON字符串	149
7.2.2	JSON对象	150
7.3	日期处理	151
7.4	正则表达式	153
7.4.1	正则表达式的定义	153
7.4.2	正则表达式相关的术语	154
7.4.3	正则表达式的语法	154
7.4.4	JavaScript中的正则表达式	156
7.4.5	正则表达式程序设计	157
7.4.6	字符串对象与正则表达式对象	158

第3部分**客户端 JavaScript****161****第8章****客户端JavaScript与HTML****162**

8.1	客户端JavaScript的重要性	162
8.1.1	Web应用程序的发展	162
8.1.2	JavaScript的性能提升	162
8.1.3	JavaScript的作用	163
8.2	HTML与JavaScript	163
8.2.1	网页显示过程中的处理流程	163
8.2.2	JavaScript的表述方式及其执行流程	163
8.2.3	执行流程的小结	166
8.3	运行环境与开发环境	166
8.3.1	运行环境	166
8.3.2	开发环境	166
8.4	调试	167
8.4.1	alert	167
8.4.2	console	167
8.4.3	onerror	169
8.4.4	Firebug, Web Inspector (Developer Tools), Opera Dragonfly	169
8.5	跨浏览器支持	171
8.5.1	应当提供支持的浏览器	171
8.5.2	实现方法	172
8.6	Window对象	174
8.6.1	Navigator对象	174
8.6.2	Location对象	174

8.6.3	History对象	175
8.6.4	Screen对象	176
8.6.5	对Window对象的引用	176
8.6.6	Document对象	176
第9章	DOM	177
9.1	DOM的定义	177
9.1.1	DOM Level 1	177
9.1.2	DOM Level 2	177
9.1.3	DOM Level 3	178
9.1.4	DOM的表述方式	178
9.2	DOM的基础	179
9.2.1	标签、元素、节点	179
9.2.2	DOM操作	179
9.2.3	Document对象	179
9.3	节点的选择	180
9.3.1	通过ID检索	180
9.3.2	通过标签名检索	180
9.3.3	通过名称检索	184
9.3.4	通过类名检索	184
9.3.5	父节点、子节点、兄弟节点	185
9.3.6	XPath	187
9.3.7	Selector API	189
9.4	节点的创建与新增	190
9.5	节点的内容更改	190
9.6	节点的删除	190
9.7	innerHTML/textContent	190
9.7.1	innerHTML	190
9.7.2	textContent	191
9.8	DOM操作的性能	191
第10章	事件	192
10.1	事件驱动程序设计	192
10.2	事件处理程序/事件侦听器的设定	192
10.2.1	指定为HTML元素的属性	193
10.2.2	指定为DOM元素的属性	194
10.2.3	通过EventTarget.addEventListener()进行指定	194
10.2.4	事件处理程序/事件侦听器内的this引用	196
10.3	事件的触发	196
10.4	事件的传播	196
10.4.1	捕获阶段	197
10.4.2	目标阶段	197
10.4.3	事件冒泡阶段	197
10.4.4	取消	197
10.5	事件所具有的元素	198
10.6	标准事件	199
10.6.1	DOM Level 2中所定义的事件	199
10.6.2	DOM Level 3中所定义的事件	200
10.7	自定义事件	202
第11章	客户端JavaScript实践	203
11.1	样式	203
11.1.1	样式的变更方法	203
11.1.2	位置的设定	207
11.1.3	位置	208
11.1.4	动画	209
11.2	AJAX	210
11.2.1	异步处理的优点	210
11.2.2	XMLHttpRequest	210
11.2.3	基本的处理流程	210
11.2.4	同步通信	212
11.2.5	超时	212
11.2.6	响应	213
11.2.7	跨源限制	214
11.2.8	跨源通信	214
11.2.9	JSONP	214
11.2.10	iframe攻击 (iframe hack)	215
11.2.11	window.postMessage	218
11.2.12	XMLHttpRequest Level 2	219
11.2.13	跨源通信的安全问题	219
11.3	表单	219

11.3.1	表单元素	219
11.3.2	表单控件	221
11.3.3	内容验证	221
11.3.4	可用于验证的事件	222
11.3.5	使用表单而不产生页面跳转的方法	222
第12章	库	224
12.1	使用库的原因	224
12.2	jQuery的特征	224
12.3	jQuery的基本概念	225
12.3.1	使用实例	225
12.3.2	链式语法	226
12.4	\$函数	227
12.4.1	抽取与选择器相匹配的元素	227
12.4.2	创建新的DOM元素	227
12.4.3	将已有的DOM元素转换为jQuery对象	227
12.4.4	对DOM构造完成后的事件侦听器进行设定	227
12.5	通过jQuery进行DOM操作	228
12.5.1	元素的选择	228
12.5.2	元素的创建·添加·替换·删除	230
12.6	通过jQuery处理事件	231
12.6.1	事件侦听器的注册·删除	231
12.6.2	事件专用的事件侦听器注册方法	232
12.6.3	ready()方法	232
12.7	通过jQuery对样式进行操作	233
12.7.1	基本的样式操作	233
12.7.2	动画	234
12.8	通过jQuery进行AJAX操作	235
12.8.1	AJAX()函数	235
12.8.2	AJAX()的包装函数	236
12.8.3	全局事件	237
12.9	Deferred	237
12.9.1	Deferred的基本概念	237
12.9.2	状态迁移	238
12.9.3	后续函数	239
12.9.4	并行处理	241
12.10	jQuery插件	241
12.10.1	使用jQuery插件	241
12.10.2	创建jQuery插件	242
12.11	与其他库共同使用	243
12.11.1	\$对象的冲突	243
12.11.2	避免\$对象的冲突	243
12.12	库的使用方法	244

第4部分 HTML5

245

第13章	HTML5概要	272
13.1	HTML5的历史	246
13.2	HTML5的现状	247
13.2.1	浏览器的支持情况	247
13.2.2	Web应用程序与原生应用程序	248
13.3	HTML5的概要	248
第14章	Web应用程序	250
14.1	History API	250
14.1.1	History API的定义	250
14.1.2	哈希片段	250
14.1.3	接口	251
14.2	ApplicationCache	255
14.2.1	关于缓存管理	255
14.2.2	缓存清单文件	255
14.2.3	ApplicationCache API	258
14.2.4	在线与离线	259
第15章	与桌面应用的协作	260
15.1	Drag Drop API	260
15.1.1	Drag Drop API的定义	260

15.1.2	接口	261
15.1.3	基本的拖动与释放	262
15.1.4	自定义显示	263
15.1.5	文件的Drag-In/ Drag-Out	265
15.2	File API	267
15.2.1	File API的定义	267
15.2.2	File对象	267
15.2.3	FileReader	269
15.2.4	data URL	271
15.2.5	FileReaderSync	273

第16章 存储

274

16.1	Web Storage	274
16.1.1	Web Storage的定义	274
16.1.2	基本操作	275
16.1.3	storage事件	277
16.1.4	关于Cookie	277
16.1.5	命名空间的管理	278
16.1.6	版本的管理	279
16.1.7	对localStorage的模拟	279
16.2	Indexed Database	280
16.2.1	Indexed Database的定义	280
16.2.2	基础架构	280
16.2.3	连接数据库	281
16.2.4	对象存储的创建	281
16.2.5	数据的添加·删除·引用	282
16.2.6	索引的创建	283
16.2.7	数据的检索与更新	284
16.2.8	数据的排序	285
16.2.9	事务	285
16.2.10	同步API	286

第17章 WebSocket

287

17.1	WebSocket概要	287
17.1.1	WebSocket的定义	287
17.1.2	现有的通信技术	287
17.1.3	WebSocket的标准	290
17.1.4	WebSocket的执行方式	290
17.2	基本操作	291
17.2.1	连接的建立	291
17.2.2	消息的收发	291
17.2.3	连接的切断	292
17.2.4	连接的状态确认	292
17.2.5	二进制数据的收发	293
17.2.6	WebSocket实例的属性一览	293
17.3	WebSocket实践	294
17.3.1	Node.js的安装	294
17.3.2	服务器端的实现	295
17.3.3	客户端的实现	295
17.3.4	客户端的实现2	296

第18章 Web Workers

298

18.1	Web Workers概要	298
18.1.1	Web Workers的定义	298
18.1.2	Web Workers的执行方式	298
18.2	基本操作	299
18.2.1	工作线程的创建	299
18.2.2	主线程一侧的消息收发	299
18.2.3	工作线程一侧的消息收发	300
18.2.4	工作线程的删除	300
18.2.5	外部文件的读取	301
18.3	Web Worker实践	301
18.3.1	工作线程的使用	301
18.3.2	中断对工作线程的处理	302
18.4	共享工作线程	304
18.4.1	共享工作线程的定义	304
18.4.2	共享工作线程的创建	304
18.4.3	共享工作线程的消息收发	305
18.4.4	共享工作线程的删除	306
18.4.5	共享工作线程的应用实例	306

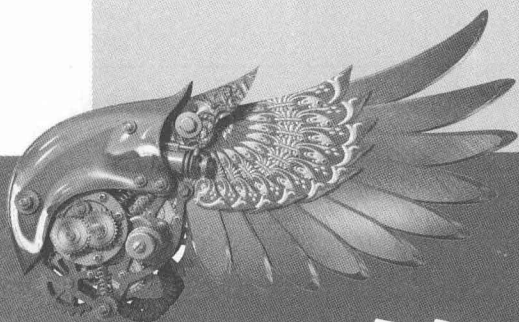
第5部分 Web API**309**

第19章	Web API的基础	310
19.1	Web API与Web服务	310
19.2	Web API的历史	311
19.2.1	Web抓取	311
19.2.2	语义网	311
19.2.3	XML	311
19.2.4	Atom	312
19.2.5	JSON	312
19.2.6	SOAP	313
19.2.7	REST	313
19.2.8	简单总结	313
19.3	Web API的组成	314
19.3.1	Web API的形式	314
19.3.2	Web API的使用	315
19.3.3	RESTful API	315
19.3.4	API密钥	316
19.4	用户验证与授权	317
19.4.1	Web应用程序的会话管理	317
19.4.2	会话管理与用户验证	318
19.4.3	Web API与权限	319
19.4.4	验证与授权	320
19.4.5	OAuth	321
第20章	Web API的实例	323
20.1	Web API的分类	323
20.2	Google Translate API	324
20.2.1	准备	325
20.2.2	执行方式的概要	325
20.2.3	使用了Web API的代码示例	326
20.2.4	微件 (Google Translate Element)	327
20.3	Google Maps API	328
20.3.1	Google Static Maps API	328
20.3.2	我的地图	329
20.3.3	Google Maps API的概要	330
20.3.4	简单的Google Maps API示例	330
20.3.5	事件	331
20.3.6	Geolocation API与Geocoding API	333
20.4	Yahoo! Flickr	334
20.4.1	Flickr Web API的使用	335
20.4.2	Flickr Web API的使用实例	336
20.5	Twitter	337
20.5.1	搜索API	337
20.5.2	REST API	338
20.5.3	Twitter JS API @anywhere	339
20.5.4	Twitter Widget	341
20.6	Facebook	341
20.6.1	Facebook应用的发展历程	341
20.6.2	Facebook的JavaScript API	343
20.6.3	Facebook的插件	344
20.7	OpenSocial	345

第6部分 服务器端 JavaScript**351**

第21章	服务器端JavaScript与Node.js	352
21.1	服务器端JavaScript的动向	352
21.2	CommonJS	352
21.2.1	CommonJS的定义	352
21.2.2	CommonJS的动向	353
21.2.3	模块功能	353
21.3	Node.js	355
21.3.1	Node.js概要	355
21.3.2	node指令	359

21.3.3	npm与包	359
21.3.4	console模块	360
21.3.5	util模块	361
21.3.6	process对象	362
21.3.7	全局对象	363
21.3.8	Node.js程序设计概要	363
21.3.9	事件API	365
21.3.10	缓冲	369
21.3.11	流	372
第22章	Node.js程序设计实践	374
22.1	HTTP服务器处理	374
22.1.1	HTTP服务器处理的基本流程	374
22.1.2	请求处理	375
22.1.3	响应处理	376
22.1.4	POST请求处理	377
22.2	HTTP客户端处理	378
22.3	HTTPS处理	379
22.3.1	通过openssl指令发布自签名证书的方法	379
22.3.2	HTTPS服务器	379
22.4	Socket.IO与WebSocket	380
22.5	下层网络程序设计	381
22.5.1	下层网络处理	381
22.5.2	套接字的定义	382
22.5.3	套接字程序设计的基本结构	382
22.5.4	套接字程序设计的具体实例	384
22.6	文件处理	385
22.6.1	本节的范例代码	385
22.6.2	文件的异步处理	386
22.6.3	文件的同步处理	386
22.6.4	文件操作相关函数	387
22.6.5	文件读取	387
22.6.6	文件写入	388
22.6.7	目录操作	389
22.6.8	对文件更改的监视	390
22.6.9	文件路径	390
22.7	定时器	390
22.8	Express	391
22.8.1	URL路由	392
22.8.2	请求处理	392
22.8.3	响应处理	393
22.8.4	scaffold创建功能	393
22.8.5	MVC架构	393
22.8.6	模板语言Jade	394
22.8.7	MongoDB (数据库)	395
22.8.8	Mongoose的实例	397
22.8.9	使用了Express与Mongoose的Web应用程序	398
	后记	401
	索引	403



第 1 部分

JavaScript 概要

本书首先介绍 JavaScript 的现状、语言特性，以及与其相关的一些领域。

第1章



JavaScript 概要

本章将介绍 JavaScript 和 ECMAScript 的关系与历史，以及 JavaScript 与作为其实现方式和运行环境的浏览器的关系，此外还将总括 JavaScript 的可移植性。

1.1

JavaScript 概要

我们首先介绍 JavaScript 相关的运行环境，其语言特征会在第2部分详述。正在读本书的读者，应该都知道 JavaScript 是在浏览器中运行的语言吧。甚至可以说，除开发者以外，被大众所熟知的程序设计语言也许只有 JavaScript。而且在软件史上，以能够在各种环境下运行而著称的语言中，大概没有比 JavaScript 更有名的了。

但是，正是由于太过常见，才让很多人对 JavaScript 有了一些误解与偏见。

例如，因为和浏览器的关联性过强，很多人都以为 JavaScript 只能在浏览器中运行。对 JavaScript 的看法也是莫衷一是。有人认为它降低了 Web 的使用体验，也有人称赞它是一门使 Web 的易用性得以进化的出色的技术。有人觉得 JavaScript 是任何人都可以学会的简单语言，也有人认为它过于抽象，很难掌握。

对 JavaScript 的看法各有不同，很难说哪一种正确。不过，只要软件以 Web 为中心，今后 JavaScript 的重要性就一定会进一步提升。JavaScript 领域的名人道格拉斯·克克福德曾把 JavaScript 称为 Web 上的虚拟机。其核心含义是，在 JavaScript 广为普及的现在，Web 已经成为了 JavaScript 事实上的运行环境。夸张地讲，JavaScript 正日益成为支配世界的程序设计语言。

虽说 JavaScript 已被逐渐应用于浏览器之外的场合，但就目前而言，其主战场还是浏览器。本书除第6部分之外，原则上将 JavaScript 作为在浏览器中运行的客户端语言。

1.2

JavaScript 的历史

JavaScript 于 1995 年登场，运用在当时最流行的浏览器 Netscape Navigator 中。在此之前，浏览器只能处理 HTML 与图片，而 JavaScript 使得浏览器端的程序运行成为可能。

能够在浏览器中运行程序，并非 JavaScript 的专利。其先驱是另一门著名的程序设计语言 Java，主要用于服务器端。当初被称为 Java Applet 的程序由于可以在浏览器（HotJava）中运行而广受瞩目。

众所周知，尽管 Java 和 JavaScript 在保留字和关键字等表层范畴上很相似，但作为程序设计语言，它们之间其实并没有什么关系。JavaScript 开发得较晚，开发之初的名称是 LiveScript，之后才决定效仿已经颇为有名的 Java，改为 JavaScript。虽然 Java 和 JavaScript 的命名导致了许多误解，但回顾历史，可以说这是一种正确的营销手段。

稍微了解一下语言规则就会发现，Java 和 JavaScript 的执行方式并不像其表面那样相似。JavaScript 反而和 Ruby 或 Python 这样的轻型脚本语言，或 Lisp 之类的以函数作为主体的程序设计语言更为相似。不过由于早期主要是跟随 Java 发展，因此 JavaScript 的对象名以及方法名和 Java 比较相似。