FULL STACK PRACTICAL

Connect Data Restructure Logic
perfect experience

full stack
practical book

全栈开发实战宝典

优逸客科技有限公司 编著

Web全栈工程师已成为DT时代下的代言人

重塑业务交互逻辑，构建完善的用户体验，打通数据和用户之间的通道

机械工业出版社
CHINA MACHINE PRESS

全栈开发实战宝典

优逸客科技有限公司　编著

机械工业出版社

“全栈”翻译自英文 Full Stack，表示为了完成一个项目所需要的一系列技术的集合。IT 行业发展到现阶段，开发一个 Web 应用，工程师需要具备的技能涵盖：前端标记语言（如 HTML 5、CSS 3）、前端编程语言（如 JavaScript）、服务器端编程语言（如 Node.js）、数据库（如 MongoDB）等，这些技术互相联系、互相依赖，缺一不可。

本书分享了全栈工程师的技能要求、核心竞争力、未来发展方向，以及对移动端的思考，内容涵盖了 Web 全栈开发的方方面面。本书既可以为互联网行业新人提供一幅精准的技术路线图，又可以作为相关从业程序员即学即用的工具书。

图书在版编目（CIP）数据

全栈开发实战宝典 / 优逸客科技有限公司编著. —北京：机械工业出版社，2018.11

ISBN 978-7-111-61262-9

Ⅰ. ①全… Ⅱ. ①优… Ⅲ. ①网页制作工具－程序设计

Ⅳ. ①TP393.092.2

中国版本图书馆 CIP 数据核字（2018）第 249885 号

机械工业出版社（北京市百万庄大街 22 号 邮政编码 100037）

策划编辑：丁 诚 责任编辑：丁 诚

责任校对：张艳霞 责任印制：孙 炜

保定市中画美凯印刷有限公司印刷

2019 年 1 月 • 第 1 版 • 第 1 次印刷

184mm×260mm • 29.5 印张 • 727 千字

0001－3000 册

标准书号：ISBN 978-7-111-61262-9

定价：99.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换

电话服务

服务咨询热线：（010）88361066

读者购书热线：（010）68326294

（010）88379203

网络服务

机 工 官 网：www.cmpbook.com

机 工 官 博：weibo.com/cmp1952

教育服务网：www.cmpedu.com

金 书 网：www.golden-book.com

前　言

读者现在拿在手里的这本书是优逸客科技有限公司出品的“实战宝典”系列丛书的第二部，第一部《HTML 5 实战宝典》可以在各大书店及相关网店购买。

优逸客科技有限公司是一家全国知名的实训机构，拥有一流的师资团队，成员大多来自北京、上海的一线公司，他们拥有丰富的实战经验。多年来，从这里走出了 3000 多名优秀的前端工程师，成就了无数学员的梦想。2014 年初，我们决定把自己掌握的知识和经验分享给更多的人，几经筹划，最终本系列丛书诞生。当时大量 Hybird App 和 Web App 兴起，企业为了追求高效，会使用 HTML 5 替代原生开发，导致市场对 HTML 5 技术人才的需求量大大增加，为了满足这个需求，我们出版了丛书的第一部《HTML 5 实战宝典》。很多前端工程师学习阅读之后纷纷表示受益匪浅，积极反馈要求分享更多的知识以适应新的 Web 开发趋势，正好在此期间我们的课程体系经过多次打磨修正，全面升级到了全端+全栈的体系，我们觉得非常有必要把全端+全栈的知识分享给大家，所以就有了丛书的第二部《全栈开发实战宝典》，希望大家喜欢！

本书内容主要围绕全栈开发展开，那么什么是“全栈”？

“全栈”翻译自英文 Full-Stack，表示为了完成一个项目所需要的一系列技术的集合。IT 行业发展到现阶段，开发一个 Web 应用，工程师需要具备的技能涵盖前端标记语言、前端编程语言、服务器端编程语言、数据库等，这些技术互相联系、互相依赖，缺一不可。以至于所有的 IT 公司都亟须有全栈人才的加入，缩短开发周期、减少开发成本、增强产品的伸缩性和可维护性。

近几年前端技术飞速发展，使得前端程序语言 JavaScript“焕发”出了它本该具有的光芒。在这种力量的支持下，后台 JavaScaript 也迅猛发展，我们称之为 Node.js。Node.js 的诞生使得前端程序员无障碍地进入到后台世界，与此同时，非关系型数据库如火如荼。JavaScript 再次发力完成对 MongoDB 的操作和控制。至此，JavaScript 以全新的姿态进入人们的视野。一种语言能够完美地衔接前端、后台、数据库，这是其他语言难以做到的，使得前端人员迅速占领了全栈工程师的高地。

围绕这两个核心概念，本书分为 4 部分来介绍全端和全栈开发中涉及的技术。

1. 全栈之 Java Script

本部分主要介绍 Java Script，包含它的“前世今生”、语法结构、操作逻辑等。这一部分会结合读者在工作中遇到的大量实例来全面剖析 Java Script 的每一个知识点。通过对比的方式对 ES5 和 ES6 做解析，让读者能够明白新的语法结构和语言升级的来龙去脉。

2. 全栈之 PHP+MySQL

本部分主要介绍 PHP 和 MySQL。作为一名前端工程师来说，向后台进军显然是不太容易的。我们需要掌握新的语言，需要掌握不同于前台的编程思想，需要和数据打交道。但幸运的是，Node.js 的诞生帮助前端人员大大降低了进入后台的门槛。但是 Node.js 作为一个“新秀”，它天然继承了大部分语言的精髓，所以入门相对来说比较难，于是我们先从 PHP

这个简单、优秀的语言说起，让读者先明白一个应用前后台的架构模式和编程思想，然后再进入 Node.js 的学习。

这一部分内容并不是本书的重点，但是它起着承前启后的作用，既能整合前面前端的内容，又能理清后面 Node.js 的编程思想，同时读者还能再掌握一门语言。这一部分中，我们将会以实际项目开发的思路带领读者写一个自己的 MVC 框架，这样既能熟悉 PHP 语法，又能了解 PHP 的主流编程思想，同时还能掌握现在流行的一些架构模式。

3. 全栈之框架

本部分对 Angular JS 和 React JS 等流行的框架进行了详细的阐述。

4. 全栈之 Node.js

Node.js 是一个让 JavaScript 运行在浏览器之外的平台。它实现了诸如文件系统、模块、包、操作系统 API、网络通信等 JavaScript Core 没有或不完善的功能。

本部分的讲解覆盖了 Web 开发中的大多数知识点，从原生 JavaScript 到 AngularJS、ReactJS 框架，从 PHP 到 Node.js，深入讲解 ES6 核心内容，全面剖析 Node.js 开发模式，全面解读 MVVM 模式和组件化开发模式，全面分析流行框架以及前端自动化开发工具的原理。选题案例应用价值高，且全部来源于大型项目的真实案例，完全可以应用于真实的项目。

同时，本书作者团队曾指导上千名前端、全栈开发工程师高薪就业，学习成果久经考验，列举的实例数量多，质量高，覆盖最前沿的技术方案。他们不仅精通程序开发，同时又是经验丰富的讲师，对学习过程中的重点、难点，以及学生容易感到困惑的点都有非常精准的把控，知识点之间的关联、顺序都是根据多年的实训经验积淀而成，这一点在本书的各个章节中都有体现。

看着这样一本书的诞生，我们百感交集，在这里要感谢所有为本书付出了大量精力的同事，没有他们的辛勤工作，就没有本书的问世，他们分别是优逸客公司总经理张宏帅，副总经理严武军，实训副总监岳英俊，全栈团队马彦龙、候宁洲、王琦、马松、李星、石晓蕾、杨晓春、杨登辉。在本书编写过程中，他们加班加点，几易其稿，精益求精，力求做到让所有知识点都清晰明了，力争每一段示例代码都是经过深思熟虑的精品，尽最大努力尝试让读者在阅读的过程中，不仅可以学到技能，同时还能感受到代码之美！

因为作者水平有限，书中难免会出现纰漏和瑕疵，请广大读者批评、指正。

编　者

目　录

前言

第 1 部分　全栈之 JavaScript ····· 1

第 1 章　JavaScript 基础概念 ····· 2
1.1　JavaScript 的用途 ····· 3
1.1.1　数据的验证 ····· 3
1.1.2　制作页面动态效果 ····· 3
1.1.3　对事件做出响应 ····· 3
1.1.4　单页面应用 ····· 3
1.1.5　网页游戏 ····· 3
1.1.6　服务器端的应用 ····· 4
1.2　JavaScript 的发展历史 ····· 4
1.2.1　悄然诞生 ····· 4
1.2.2　稳步发展 ····· 5
1.2.3　黄金时代 ····· 5
1.2.4　JavaScript 和 ECMAScript ····· 5
1.2.5　JavaScript 和 Java ····· 5
1.3　JavaScript 的语法特点 ····· 6
1.3.1　基于对象 ····· 6
1.3.2　事件驱动 ····· 6
1.3.3　松散型 ····· 7
1.3.4　解释型 ····· 7
1.4　JavaScript 的引入方式 ····· 7
1.4.1　在域名或者重定向的位置引入 ····· 7
1.4.2　在事件中引入 ····· 8
1.4.3　在页面中嵌入 ····· 8
1.4.4　引入外部 JavaScript 文件 ····· 9
1.4.5　注意事项 ····· 9
1.5　JavaScript 中的输出工具 ····· 9
1.5.1　console ····· 10
1.5.2　alert() ····· 10
1.5.3　document.write(); ····· 11
1.5.4　prompt(str,[value]); ····· 11
1.5.5　confirm() ····· 11
1.5.6　JavaScript 注释 ····· 12
第 2 章　基本构成 ····· 13
2.1　JavaScript 变量 ····· 14
2.1.1　变量的概念 ····· 14
2.1.2　变量的声明和赋值 ····· 15
2.1.3　声明变量的其他注意事项 ····· 16
2.2　数据类型 ····· 17
2.2.1　typeof 操作符 ····· 18
2.2.2　初始类型 ····· 18
2.2.3　引用类型 ····· 21
2.3　JavaScript 运算符 ····· 21
2.3.1　算术运算符 ····· 22
2.3.2　关系运算符（或比较运算符） ····· 23
2.3.3　赋值运算符 ····· 25
2.3.4　逻辑运算符 ····· 25
2.3.5　一元运算符 ····· 27
2.3.6　三元运算符 ····· 29
2.3.7　特殊运算符 ····· 29
2.4　JavaScript 流程控制 ····· 30
2.4.1　名词解释 ····· 30
2.4.2　选择结构 ····· 30
2.4.3　循环结构 ····· 35
第 3 章　函数和数组 ····· 39
3.1　函数的基本概念 ····· 40

3.1.1 函数的声明 40
3.1.2 函数的调用 41
3.1.3 参数 44
3.1.4 函数的返回值 46
3.1.5 作用域 47
3.1.6 回调函数 49
3.1.7 递归函数 50
3.1.8 闭包函数 51
3.2 内置顶层函数和数据类型转换 52
3.2.1 内置顶层函数 52
3.2.2 数据类型转换 53
3.3 ES6 中新增的函数语法 54
3.3.1 函数参数的默认值 54
3.3.2 函数的 name 属性 55
3.3.3 箭头函数 55
3.4 数组 56
3.4.1 数组的概念 56
3.4.2 数组的创建 56
3.4.3 数组的访问 58
3.4.4 数组的遍历 58
第 4 章 对象 61
4.1 JavaScript 对象 62
4.1.1 名词解释 62
4.1.2 创建对象的方法 63
4.1.3 属性与方法 64
4.1.4 销毁对象 65
4.1.5 对象的遍历 66
4.1.6 对象的存储方式 66
4.1.7 instanceof 66
4.2 对象的特性 67
4.2.1 对象的特性——封装 67
4.2.2 对象的特性——继承 68
4.2.3 this 指针 70
4.2.4 对象的分类 72
4.3 ES6 中对象的新特性 72
4.3.1 类的支持 72
4.3.2 变量的解构赋值 73
4.3.3 扩展运算符（spread）和 rest 参数 74
4.3.4 属性的简洁表示 76
4.3.5 属性名表达式 77
4.3.6 方法的 name 属性 77
第 5 章 原生对象 78
5.1 Object 对象 79
5.1.1 Object 的常用方法 79
5.1.2 属性的遍历（Object 对象方法的使用） 83
5.2 Math 对象 84
5.2.1 Math 对象的属性 84
5.2.2 Math 对象的方法 84
5.3 字符串对象 86
5.3.1 创建 String 对象 86
5.3.2 字符串对象的属性 86
5.3.3 字符串对象的方法 86
5.4 数组对象 91
5.4.1 数组对象的属性 91
5.4.2 数组对象的方法 92
5.4.3 数组对象的构造函数的方法 98
5.5 日期对象 99
5.5.1 定义日期对象 99
5.5.2 获取日期信息的方法 99
5.5.3 设置日期的方法 100
5.6 正则 101
5.6.1 正则表达式的概念 101
5.6.2 应用场合 102
5.6.3 创建正则表达式 102
5.6.4 正则表达式的模式 103
5.6.5 正则方法 104
5.6.6 字符串中用到正则的函数 106
5.7 Set 数据结构 107
5.7.1 Set 基本用法 107
5.7.2 Set 属性和方法 108
5.7.3 Set 遍历方法 108
5.7.4 WeakSet 109
5.8 Map 数据结构 109

5.8.1 Map 基本用法 …… 109
5.8.2 Map 属性和方法 …… 110
5.8.3 Map 遍历方法 …… 111
5.8.4 Map 与数组对象的转换 …… 112
5.8.5 WeakMap …… 113
第 6 章 常见网页效果制作 …… 114
6.1 BOM 介绍 …… 115
6.1.1 window 对象 …… 115
6.1.2 document 对象 …… 117
6.1.3 history 对象 …… 118
6.1.4 location 对象 …… 118
6.1.5 screen 对象 …… 119
6.1.6 navigator 对象 …… 119
6.2 DOM 介绍 …… 119
6.2.1 对内容进行操作 …… 121
6.2.2 对样式进行操作 …… 122
6.2.3 对属性的操作 …… 123
6.2.4 对类名的操作 …… 123
6.2.5 事件 …… 124
6.2.6 综合运用——制作网页轮播图效果 …… 128
6.2.7 获取位置和尺寸 …… 133
6.2.8 获取具有滚动条元素的滚动位置 …… 134
6.2.9 案例展示——楼层跳转效果制作 …… 135
6.2.10 结点的属性和方法 …… 137
6.2.11 事件对象 …… 138
6.2.12 事件流 …… 142
6.2.13 案例展示——移动端可拖曳轮播图展示 …… 144
6.3 综合练习——面向对象的打字游戏 …… 146
第 7 章 AJAX 详解 …… 154
7.1 AJAX 原理介绍 …… 155
7.1.1 AJAX 的特点 …… 155
7.1.2 与传统的 Web 应用比较 …… 155
7.1.3 AJAX 的工作原理 …… 155
7.1.4 XMLHttpRequest 对象 …… 155
7.1.5 GET 和 POST 的区别 …… 157
7.1.6 同步和异步的区别 …… 158
7.2 AJAX 函数封装 …… 158
7.3 AJAX 运用 …… 160
第 8 章 客户端存储及应用 …… 165
8.1 Cookie 简介 …… 166
8.1.1 Cookie 的作用 …… 166
8.1.2 Cookie 的基本概念 …… 166
8.1.3 Cookie 的用法 …… 167
8.1.4 Cookie 的封装函数 …… 167
8.1.5 利用 Cookie 保存文字阅读器的状态 …… 168
8.2 localStorage 和 sessionStorage 简介 …… 170
8.3 使用 localStorage …… 172
第 9 章 jQuery 原理及用法 …… 178
9.1 jQuery 概述 …… 179
9.2 jQuery 核心思想 …… 179
9.3 jQuery 隐式循环 …… 181
9.4 jQuery 链式调用 …… 183
9.5 jQuery 跨平台 …… 184
9.6 jQuery 选择器 …… 186
9.7 jQuery 筛选 …… 187
9.7.1 过滤 …… 187
9.7.2 查找 …… 187
9.7.3 串联 …… 188
9.8 jQuery 属性 …… 188
9.8.1 属性 …… 188
9.8.2 CSS 类 …… 189
9.8.3 HTML 代码/文本/值 …… 189
9.9 jQuery CSS …… 189
9.9.1 样式 …… 190
9.9.2 位置 …… 190
9.9.3 尺寸 …… 191
9.10 jQuery 文档处理 …… 191
9.10.1 内部插入 …… 192
9.10.2 外部插入 …… 192

9.10.3 包裹 …… 193
9.10.4 替换 …… 193
9.10.5 删除 …… 193
9.10.6 复制 …… 194
9.11 jQuery 事件 …… 195
9.11.1 页面载入 …… 195
9.11.2 事件处理 …… 195
9.11.3 事件触发 …… 196
9.11.4 事件委派 …… 196
9.11.5 事件 …… 196
9.12 jQuery 事件对象 …… 197
9.13 jQuery 效果 …… 198
9.13.1 基本方式 …… 198
9.13.2 自定义动画 …… 199
9.13.3 动画控制 …… 199
9.14 jQuery AJAX …… 200
9.14.1 json 参数的选项 …… 200
9.14.2 AJAX 的函数实现 …… 200
9.14.3 全局处理函数 …… 201
9.15 jQuery 工具 …… 202
9.16 综合案例制作——轮播图 …… 204
9.17 综合案例制作——扑克牌 …… 206

第 2 部分 全栈之 PHP+MySQL …… 212

第 10 章 PHP 基础 …… 213
10.1 PHP 的使用 …… 214
10.2 PHP 的数据类型 …… 216
10.3 PHP 的变量 …… 218
10.4 PHP 的常量 …… 219
10.5 PHP 的表达式、运算符和流程控制 …… 220
10.6 PHP 的函数 …… 221
10.7 PHP 的类与对象 …… 223
10.8 PHP 使用 PDO 连接数据库 …… 227
第 11 章 MySQL 基础 …… 230
11.1 MySQL 简介 …… 231
11.2 检索数据 …… 231
11.3 排列数据 …… 233
11.4 过滤数据 …… 234
11.5 计算字段 …… 236
11.6 使用函数 …… 236
11.7 分组数据 …… 238
11.8 联结表 …… 239
11.9 插入数据 …… 240
11.10 更新和删除数据 …… 240
11.11 创建和操作表 …… 240
11.12 使用视图 …… 241
第 12 章 PHP 框架 …… 243

第 3 部分 全栈之框架 …… 251

第 13 章 AngularJS …… 252
13.1 AngularJS 简介 …… 253
13.2 AngularJS 特性 …… 253
13.3 AngularJS 核心思想 …… 253
13.4 AngularJS 的优势 …… 254
13.5 AngularJS 应用组成 …… 254
13.6 AngularJS 环境搭建 …… 254
第 14 章 第一个应用程序 …… 255
14.1 AngularJS MVC 架构 …… 256
14.2 AngularJS 应用实例 …… 256
14.2.1 原生 JavaScript 实现 …… 256
14.2.2 AngularJS 实现 …… 256
第 15 章 AngularJS 模块 …… 258
15.1 AngularJS 模块简介 …… 259

15.2 模块的优点 …… 259
15.3 创建模块 …… 259
15.4 添加控制器 …… 259
15.5 添加指令 …… 260
第 16 章 作用域 …… 261
16.1 作用域简介 …… 262
16.1.1 作用域的作用 …… 262
16.1.2 作用域概述 …… 262
16.2 定义作用域 …… 262
16.3 作用域层级 …… 263
第 17 章 控制器 …… 265
17.1 控制器简介 …… 266
17.2 控制器定义 …… 266
17.2.1 函数式创建 …… 266
17.2.2 模块化定义 …… 266
17.3 控制器嵌套 …… 267
第 18 章 表达式 …… 268
18.1 表达式概述 …… 269
18.1.1 表达式简介 …… 269
18.1.2 AngularJS 表达式的特性 …… 269
18.2 表达式的使用 …… 269
18.2.1 表达式数字 …… 269
18.2.2 表达式字符串 …… 269
18.2.3 表达式对象 …… 270
18.2.4 表达式数组 …… 270
第 19 章 过滤器 …… 271
19.1 过滤器简介及其用法 …… 272
19.2 在 HTML 中使用过滤器 …… 272
19.2.1 currency 过滤器 …… 272
19.2.2 lowercase/uppercase 过滤器 …… 273
19.2.3 filter 过滤器 …… 273
19.2.4 limitTo 过滤器 …… 273
19.2.5 orderBy 过滤器 …… 274
19.3 在 JavaScript 中使用过滤器 …… 275
19.4 自定义过滤器 …… 275
第 20 章 指令 …… 276
20.1 指令简介 …… 277
20.2 内置指令 …… 277
20.2.1 基础指令 …… 277
20.2.2 事件类指令 …… 277
20.2.3 表单类指令 …… 278
20.2.4 样式类指令 …… 279
20.2.5 DOM 操作相关指令 …… 280
20.2.6 指令扩展 …… 280
20.3 自定义指令 …… 281
20.3.1 第一个自定义指令 …… 281
20.3.2 参数详解 …… 281
第 21 章 多重视图和路由 …… 288
21.1 路由简介 …… 289
21.2 安装 …… 289
21.3 布局 …… 289
21.4 配置 …… 290
第 22 章 依赖注入简介及使用 …… 292
第 23 章 服务 …… 296
23.1 服务概述 …… 297
23.2 使用服务 …… 297
23.2.1 $http 服务 …… 297
23.2.2 $timeout 服务 …… 299
23.2.3 $interval 服务 …… 299
23.2.4 $location 服务 …… 299
第 24 章 动画 …… 301
24.1 动画概述 …… 302
24.2 动画的安装与原理 …… 302
24.2.1 安装 …… 302
24.2.2 原理 …… 302
24.3 动画实现 …… 303
24.3.1 CSS 3 过渡 …… 303
24.3.2 CSS 3 动画 …… 304
24.3.3 JavaScript 中的动画 …… 305
24.3.4 在定义指令中使用动画 …… 306
第 25 章 综合案例 …… 307
第 26 章 初识 React …… 315
26.1 前端架构的变迁 …… 316
26.1.1 前后端不分的时代 …… 316
26.1.2 后端 MVC 时代 …… 316
26.1.3 AJAX 的 SPA 时代 …… 316

26.1.4 前端为主的 MV*时代 …… 317
26.2 React 简介 …… 317
26.3 专注于视图 …… 317
26.4 React 解决的问题 …… 317
26.5 虚拟 DOM 机制 …… 317
26.5.1 虚拟 DOM 介绍 …… 317
26.5.2 虚拟 DOM 的意义 …… 318
26.6 React 组件 …… 318
26.6.1 组件化的概念 …… 318
26.6.2 组件化的意义 …… 319
第 27 章 React 基础 …… 320
27.1 React 开发环境 …… 321
27.1.1 安装 React …… 321
27.1.2 浏览器端的使用 …… 321
27.1.3 基于 Webpack 工程化构建工具 …… 322
27.2 React 组件构建 …… 325
27.2.1 渲染组件 …… 325
27.2.2 创建组件 …… 326
27.2.3 示例 …… 326
27.3 JSX 简介 …… 327
27.3.1 JSX 的由来 …… 327
27.3.2 JSX 介绍 …… 327
27.3.3 使用 JSX 的原因 …… 327
27.3.4 JSX 的基本原理 …… 328
27.4 开始使用 JSX …… 328
27.4.1 在浏览器中使用 JSX …… 328
27.4.2 基本语法 …… 329
27.4.3 元素类型 …… 329
27.4.4 元素属性 …… 330
27.4.5 JSX 注释 …… 330
27.4.6 求值表达式 …… 331
27.5 React 中的非 DOM 属性 …… 332
27.5.1 dangerouslySetInnerHTML 属性 …… 333
27.5.2 ref 属性 …… 333
27.5.3 key 属性 …… 334
27.6 组件的生命周期 …… 334
27.6.1 初始化阶段 …… 335
27.6.2 运行中阶段 …… 336
27.6.3 销毁阶段 …… 336
27.7 React 操作 DOM …… 337
27.7.1 ref 获取 DOM …… 337
27.7.2 findDOMNode 获取 DOM …… 337
27.8 React 属性与状态 …… 338
27.8.1 数据流 …… 338
27.8.2 props 属性 …… 338
27.8.3 状态 …… 340
27.9 React 事件 …… 341
27.9.1 React 事件处理 …… 342
27.9.2 React 合成事件 …… 342
27.9.3 React 支持事件 …… 342
27.9.4 事件中的 this 指向 …… 342
27.9.5 在 React 中使用原生事件 …… 345
27.9.6 对比 React 合成事件与 JavaScript 原生事件 …… 345
27.10 React 表单 …… 345
27.10.1 表单类型 …… 346
27.10.2 无约束表单 …… 346
27.10.3 约束性组件 …… 346
27.10.4 表单控件 …… 347
27.11 使用 CSS 样式 …… 350
27.11.1 行内方式添加样式 …… 350
27.11.2 类名方式添加样式 …… 351
第 28 章 React 进阶 …… 352
28.1 React 组件组合 …… 353
28.1.1 组件组合 …… 353
28.1.2 组件包含 …… 355
28.2 React 组件间通信 …… 356
28.2.1 父组件向子组件通信 …… 357
28.2.2 子组件向父组件通信 …… 357
28.3 React 性能优化 …… 358
28.4 React 动画 …… 359
28.5 React Router …… 361
28.5.1 路由的基本原理 …… 361
28.5.2 React Router 特性 …… 361

28.5.3 安装 react-router-dom ····· 362
28.5.4 路由中的组件 ····· 362
28.5.5 路由实例 ····· 363
第 29 章 React 应用实例 ····· 366
29.1 项目介绍 ····· 367
29.2 项目分析 ····· 368
29.2.1 公共部分组件 ····· 368
29.2.2 首页组件划分 ····· 368
29.2.3 城市选择页面 ····· 369
29.3 准备开发环境 ····· 369
29.3.1 建立项目文件夹 ····· 369
29.3.2 项目初始化 ····· 369
29.3.3 安装指定 npm 包 ····· 370
29.3.4 Webpack 配置文件 ····· 371
29.3.5 应用目录结构 ····· 374
29.4 天气应用 ····· 375
第 4 部分 全栈之 Node.js ····· 383
第 30 章 初识 Node.js ····· 384
30.1 Node.js 简介 ····· 385
30.2 Node.js 的发展 ····· 385
30.3 Node.js 的特性 ····· 385
30.4 Node.js 的使用场景 ····· 385
30.5 Node.js 和 JavaScript 的区别与联系 ····· 386
30.6 CommonJS ····· 386
第 31 章 Node.js 的安装 ····· 388
31.1 下载 Node.js ····· 389
31.2 Node.js 的版本信息 ····· 389
31.3 Node.js 的安装方法 ····· 389
第 32 章 Node.js 模块系统 ····· 391
32.1 模块化编程 ····· 392
32.2 Node.js 中模块的分类 ····· 392
32.3 模块操作 ····· 392
32.3.1 require ····· 392
32.3.2 exports ····· 393
32.3.3 module ····· 393
32.3.4 module 的其他 API ····· 394
32.4 模块加载的优先级 ····· 394
32.5 模块路径解析规则 ····· 395
32.6 Node.js 包 ····· 396
32.6.1 包的概念 ····· 396
32.6.2 自定义包 ····· 396
32.6.3 包配置文件 package.json ····· 397
第 33 章 Node.js 包管理工具 ····· 399
33.1 npm 简介 ····· 400
33.2 npm 常见的使用场景 ····· 400
33.3 npm 常用命令 ····· 400
33.3.1 使用 npm 安装模块 ····· 400
33.3.2 全局安装与本地安装 ····· 401
33.3.3 模块中的其他操作 ····· 402
33.3.4 版本号 ····· 402
33.4 向 npm 服务器发布自己的包 ··· 402
第 34 章 Node.js 全局对象 ····· 404
34.1 Buffer 类 ····· 405
34.2 console 模块 ····· 409
34.3 process 对象 ····· 411
34.4 global 对象的方法 ····· 413
34.5 魔术常量 ····· 414
第 35 章 Node.js 常用模块 ····· 415
35.1 path 模块 ····· 416
35.2 child_process 模块 ····· 417
35.3 url 模块 ····· 419
35.4 querystring 模块 ····· 423
第 36 章 Node.js 中的 fs 模块 ····· 424
36.1 使用 fs 模块 ····· 425
36.2 常用操作 ····· 426
36.2.1 异步读取文件 ····· 426

36.2.2 同步读取文件 …… 428
第 37 章 Node.js 流 …… 434
37.1 Stream 的作用 …… 435
37.2 读取流 …… 435
37.3 写入流 …… 436
第 38 章 Node.js 中的 http 模块 …… 437
38.1 HTTP 简介 …… 438
38.2 http 模块 …… 439
第 39 章 Node.js 实战之静态服务器 …… 444
39.1 非目录文件处理 …… 445
39.2 目录处理 …… 445
39.3 实现静态服务器 …… 446
第 40 章 Node.js 实战之爬虫系统 …… 452
40.1 爬虫系统流程 …… 453
40.2 布隆过滤器 …… 455
40.3 数据存储设计 …… 457
40.4 爬虫主程序 …… 458

第 1 部分　全栈之 JavaScript

从 Web 的发展史来看，Web 1.0 以展示和呈现信息为主。从 Web 2.0 开始以使用和交换信息为主，即人机交互。交互的出现使得人们能在互联网上进行各种各样的活动，如社交、购物、观影等，而互联网的发展不仅于此。移动端的出现使得人们不再局限于计算机旁，每个人则变为行走的数据。离开了固定局促的计算机旁，人们发现基于互联网能做更多的事情，如移动支付、移动办公、移动餐饮、移动学习、移动娱乐，越来越多的事情都能在移动中完成。所以人们需要的交互越来越多，交互的逻辑也越来越复杂。于是开始进入 Web 3.0 时代也就是万物互联的时代。

JavaScript “生于” Web 1.0，但是在 Web 2.0 的时候 “粉墨登场”，它的出现正好契合了 Web 2.0 时代的标志。它使得前端页面具有了动态的效果，具有了交互的生命和产品的灵魂。于是 JavaScript 受到了越来越多的从业人员的追捧和喜爱。JavaScript 一路追随着互联网的发展在更新和迭代，于是 ES5、ES6、ES7 一路 “高奏凯歌”，全面占领了前端的高地，能够高效地运行在 PC 端、移动端、多端，为下一步布局做好充分的准备。

JavaScript 现在已然是前端的代名词，成为前端的核心技术。它能够对重构的页面进行动态操作，能够实现人类对于前端页面操作的所有想象，能够为静态的页面注入活力四射的数据。本章将对 JavaScript 展开详细的讲解，结合大量的实例来全面剖析 JavaScript 的每一个知识点。通过对比的方式对 ES5 和 ES6 做解析，让读者明白新的语法结构和语言升级的来龙去脉。

JavaScript 是进阶 Web 全栈开发的必经之路，也是 Web 全栈开发中最核心的语言，本部分对后续内容有着举足轻重的作用。

第 1 章

JavaScript 基础概念

本章主要内容

- **JavaScript 的用途**
- **JavaScript 的发展历史**
- **JavaScript 的语法特点**
- **JavaScript 的引入方式**
- **JavaScript 中的输出工具**

全端意味着利用相关的技术、编程手段、编程思想将视觉和交互效果无缝、完美地重构到多种终端。本章将从基本语法开始，详细阐述能够实现交互效果的 JavaScript。

JavaScript 是一门跨平台、面向对象的弱类型的轻量级的解释型语言，也是目前最流行的网页前端脚本语言之一，通常被称为 JS。那些炫酷的页面效果、良好的交互体验、表单验证等，都是通过 JavaScript 实现的。

随着 AJAX 的出现，前端可以在不刷新页面的情况下和后台进行数据交换，从而实现页面数据的更新。jQuery 库的出现，使得 JS 编写变得非常简洁。类似于 ECharts、D3 插件的出现，使得前端实现丰富的数据可视化图表变得更加容易，AngularJS、React 等优秀框架的出现，大大提高了开发效率。最终 Node 的发布，使得 JS 不仅可以运行在前端，还可以运行在服务器上。现在，前端工程师通过 JS 实现网站开发已经成为现实了。

以上只是陈列了本书中涉及的一些插件和框架，当然还远不止这些，那么面对这么多框架和插件，该如何快速上手呢？既然它们都是建立在 JavaScript 的基础上，那么只要把 JavaScript 中的基本知识以及原理都理解透彻，学习将会变得容易很多。

关于 JavaScript，大家必须深刻理解并熟练运用的有 3 部分，即变量、对象、函数。具体内容会在接下来的章节中讲到。

本章将重点阐述 JavaScript 的基础概念。

1.1 JavaScript 的用途

JavaScript 作为前端开发中的核心语言，其重要性不言而喻，那么用它能做些什么呢？本节将介绍 JavaScript 的一些应用场合。

1.1.1 数据的验证

JavaScript 最初设计的目的就是用来完成表单数据的验证。到今天，利用 JavaScript 完成数据验证依然是一种重要的验证方式，当然，除了使用 JavaScript，表单中也有自带的验证功能，但是一些需要结合数据库的验证依然需要通过后台语言来完成。

例如，优逸客官网（www.sxuek.com），首页中有在线咨询的功能，其中有输入手机号的一个输入框，当用户输入信息时就需要进行数据验证。还有人们经常操作的用户登录、注册等也都需要进行数据验证。

1.1.2 制作页面动态效果

随着网络链接速度的提升和硬件设备的发展，人们越来越不满足于简单、死板的内容呈现方式，通过浏览器，用户更想看到一些不一样的效果，而 JavaScript 就是在网页中制作动态效果的比较好用的工具。

例如，优逸客官网的轮播图和楼层跳转等动态效果都是通过 JS 实现的。

1.1.3 对事件做出响应

JavaScript 是基于事件驱动的，用户可以通过单击、鼠标指针的移入移出、滚轮滚动、键盘按下等一系列的事件来控制页面中代码的执行，进而提升网页的交互性。

例如，在优逸客官网中通过单击 banner 图的轮播点、导航栏等，可以进行内容的切换等，包括鼠标指针移入后效果的变化，其实都是 JavaScript 在对事件做响应。

1.1.4 单页面应用

随着 AngularJS 等前端框架的兴起，JavaScript 能够在前端网页中处理的逻辑也更加复杂，WebStorage 更是给开发人员提供了直接在浏览器中保存数据的便利。以前一些 C/S 结构的应用现在也可以基于浏览器来实现了。

例如，谷歌在线的 Word、Excel 等编辑器；各大平台的云，如小米云、谷歌云等都是单页面应用。

1.1.5 网页游戏

H5 中的 Canvas 给开发人员提供了在页面中处理复杂 2D、3D 效果的接口，当然，操作

这些接口的还是 JavaScript。虽然如今在浏览器端的网页游戏更多的是通过其他语言实现的，但是基于 JavaScript 的游戏，现在也可以在游戏市场分一杯羹。

1.1.6 服务器端的应用

Node.js（有关 Node.js 详见本书的第 3 部分）就是运行在服务器端的 JavaScript。Node.js 是一个事件驱动 I/O 服务端的 JavaScript 环境，基于 Google 的 V8 引擎，执行 JavaScript 的速度非常快，性能非常好。

JavaScript 发展到目前为止，它不仅仅作为一种客户端语言，而且还能构建服务器，但是无论如何，学习它都需要学习以下 3 部分：

1）ECMAScript，核心语法部分。

2）浏览器对象模型（BOM），提供访问和操作网页内容的方法和接口。

3）文档对象模型（DOM），提供与浏览器交互的方法和接口。

注：在本书 Node 部分还有关于 JavaScript 的其他操作，如操作文件、操作数据库等，相关内容请查阅本书的第 3 部分。

有关 BOM 和 DOM 的详细介绍请参考第 6 章。

1.2 JavaScript 的发展历史

如上节介绍所说，JavaScript 拥有诸多用途，那么我们不禁好奇，如此强大的一门语言，它是怎么由来的呢？

1.2.1 悄然诞生

1995 年，Netscape （网景）公司的 Brendan Eich（布兰登·艾奇）（见图 1-1）在公司提出的“看上去与 Java 足够相似，但是比 Java 简单，使得非专业的网页开发者也能很快上手”的要求下，利用 10 天时间就把 JavaScript 设计出来了。当然，起初并不是 JavaScript 这个名字，最开始叫作 liveScript，因为 Sun 公司（Java 开发者所在公司）与网景合作的原因，故改名为 JavaScript。

图 1-1

同年，不甘落后的微软在自己的 IE 浏览器中嵌入了 JavaScript 的复刻版并且将其命名为 JScript，独立发展自己的客户端脚本语言。

1997 年，通过网景、Sun、微软等公司以及众多开发者的努力，统一标准的 ECMAScript 被作为标准规范推出，标准编号为 ECMA-262，从此 ECMAScript 就成为 JavaScript 等脚本语言实现的标准基础。因为命名版权的原因，JavaScript 的正式名称为“ECMAScript”。ECMA（欧洲计算机制造联合会）是制定计算机标准规范的机构，不