



Flux Architecture

Flux架构

学习使用 Flux 构建强大的、可扩展的应用，
该架构每天服务于亿万 Facebook 用户

[加]Adam Boduch 著
段金辰 马雪琴 李胜 马飞 等译

Flux Architecture

Flux架构

[加]Adam Boduch 著
段金辰 马雪琴 李胜 马飞 孙辉 译

电子工业出版社

Publishing House of Electronics Industry

北京•BEIJING

内 容 简 介

Flux 是一套架构模型，将 Web 应用的各个主要功能以组件的形式进行划分，并进一步划分子组件。而各组件又以动作、存储器和视图来进行架构分层。整体采用单向数据流的形式进行事件的响应，各组件间也强制按照单向数据流进行相互影响，直至数据流结束。在本书中，先向大家介绍了 Flux 是什么，以及简单展示了其基本构建模式。然后从动作、存储器、视图、分发器等核心概念，更为详细地阐述了 Flux 的架构模式。最后，介绍了 Flux 库、测试工具和其对其他相关技术栈的影响。

本书适用于前端开发者，以及希望对 Flux 架构有深入了解的人群。

Copyright © Packt Publishing 2016. First published in the English language under the title 'Flux Architecture'.

本书简体中文版专有出版权由Packt Publishing授予电子工业出版社。未经许可，不得以任何方式复制或抄袭本书的任何部分。专有出版权受法律保护。

版权贸易合同登记号 图字：01-2016-7233

图书在版编目（CIP）数据

Flux 架构/（加）亚当·博达哈（Adam Boduch）著；段金辰等译. —北京：电子工业出版社，2017.7
书名原文：Flux Architecture
ISBN 978-7-121-31600-5

I. ①F… II. ①亚… ②段… III. ①软件设计 IV. ①TP311.5

中国版本图书馆 CIP 数据核字（2017）第 120669 号

策划编辑：张春雨

责任编辑：刘 舫

印 刷：三河市良远印务有限公司

装 订：三河市良远印务有限公司

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：787×980 1/16

印张：20.5

字数：410 千字

版 次：2017 年 7 月第 1 版

印 次：2017 年 7 月第 1 次印刷

定 价：89.00 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）88254888，88258888。

质量投诉请发邮件至 zltts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

本书咨询联系方式：010-51260888-819 faq@phei.com.cn。

感谢 Melissa 对我全部的爱和支持。

感谢 Jason、Simon 和 Kevin 照亮我的每一天。

译者序

自 1996 年网景通信公司（Netscape Communications Corporation）将 JavaScript 提交至欧洲计算机制造联合会（ECMA）以来，整个 Web 前端应用的架构和开发发生了翻天覆地的变化，人们为了提高开发效率和带来更多新的特性，创造出了许多新的概念和工具。在很长一段时间里，MVC 及其变种或改进版本（MV*），成为 Web 前端应用中非常普遍的架构模式，其对视图和数据进行了分离但同时又保持互动，并将这种模式做得淋漓尽致，从而为高效开发 Web 应用打下基础，而且在实际生活中，为人们带来了许多杰出的产品和优秀的体验。

然而，随着时代的发展和科技的进步，人们逐渐发现当前流行的架构仍旧存在弊端，而且有些是从根本上存在的。这些弊端有时非常隐蔽，甚至来自于这些架构中所引以为豪的部分。例如，过度的关注点分离，可能会造成某项功能由支离破碎的一些组件构成，这很容易给代码造成混乱，反而不利于扩展和维护。为了解决这些问题，Flux 应运而生。2014 年，Facebook 在 F8 开发者峰会上，发表了《Flux——搭配 React 的一套应用架构》的主题演讲，从而让 Flux 受到更多人的关注。Flux 一词在拉丁文中是“流”（Flow）的意思，这个单词诠释了 Flux 的精髓，即基于单向数据流而构建的架构模型。

在 Flux 架构中，数据流从开始到结束，从一个组件到另一个组件，从行为的触发到界面的展现，都是单向的，非常直观，也便于回溯；整体架构清晰简明，一切都是基于动作、分发器、存储器和视图进行的，分层也相对简单，避免了嵌套；对信息的操作和展示，都是基于操作和获取状态来实现的，这简化了信息数据和视图间交互的流程。所有这一切，对于搭建大型 Web 应用来说，都是非常有用的，因为它能将问题变得更为简单可控，并在许多层面上保持了一致性。

由于 Flux 本身是一套架构模式，而非一套完整的框架，因此，其更多的是提供一些在架构和开发过程中的程序设计思路，这使我们能够以更有利于建设可扩展的大型 Web 应用的方式，对项目进行设计和实现，并能利用相关测试工具进行问题排查，从而在保证质量和体验的前提下，快速完成任务，并预留足够的扩展能力。当然，为了能够给开发者带来更多的便利，Flux 也相应地提供了一定量的辅助库，可以直接使用，从而避免去实现一些在该架构模式中已经规格化的功能，本书就是以此为基础来介绍 Flux 的使用的。但如果需要一套完整的基于 Flux 的库，还可以使用 Alt.js 或 Redux，前者完全遵循了 Flux 的架构理念，而后者则更多的是借鉴。利用这些库，我们可以更为快速地开发自己的项目。

事实上，Flux 并不依赖 React。但随着 React 和 Redux 的风靡，人们其实已经开始感受到 Flux 对实际项目所产生的影响，并从中获益。然而，相比较于去熟悉如何使用一套框架来说，真正能给自己带来技术上质的成长突破的，应该是更多更深地去探究这些框架的实现方式，以及其所包含的思想。通过阅读本书，大家可以从零开始，对 Flux 架构进行非常充分和完整的了解，从而能够在实际项目中，更好地运用自己所掌握的技术来提高质量和效率，并推动自身能力的进步。

段金辰

2017 年 1 月

关于作者

Adam Boduch 从事大规模 JavaScript 开发已经将近 10 年。在转行前端之前，他曾使用 Python 和 Linux 编写过几个大规模的云计算产品。Adam 对复杂度有一些了解，在真实世界的软件系统和应对其扩展上的挑战方面具有实际经验。

他写过几本 JavaScript 的书，包括《JavaScript 并发》(*JavaScript Concurrency*)，他在研究用户体验创新和高性能方面充满热情。

关于审校者

August Marcello III 是一位狂热的软件工程师，他在企业软件设计、实现和部署现代 Web 应用程序架构方面拥有近二十年的经验。他专注于整个基于 SaaS 的 Web 生态系统，取得了令人信服的用户体验，同时也证明了自己。他对新兴技术的热情，加上特别关注前沿的 JavaScript 平台，一直是他追求技术卓越的主要动力。他的业余生活也十分丰富，跑步、骑山地车以及和家人朋友待在一起。

非常感谢 Chuck、Mark、Eric 和 Adam，我有幸与他们合作并学习。同时我也非常感谢家人、朋友和我在一起的珍贵经历。

关于译者

本书由段金辰、马雪琴、李胜、马飞、孙辉翻译。

段金辰从事软件开发大约 10 年，曾供职于微软、阿里巴巴等世界知名公司，参与或主持过多项大型软件系统的架构和开发，涉及基础类库、云、Web 前端、On-premises 服务、App 等众多方向，精通包括 JavaScript 在内的多种编程语言。

马雪琴，研究生就读于华中科技大学通信工程专业，其间有幸接触到前端、数据可视化等方向，并且对这些技术感到相见恨晚。现就职于阿里巴巴集团客户体验事业群前端开发团队。平时喜欢宅着看书，学着画画，更喜欢去外面走走看看，安静而不安分。

李胜，前端开发工程师。就职于阿里巴巴集团客户体验事业群前端开发团队，曾在饿了么大前端部门参与实习，热衷于追寻前沿技术。除前端相关技术以外，对游戏开发，Go 以及其他后端技术也有一定了解。平时的爱好有动漫和科幻。

马飞，阿里巴巴前端工程师，半路出家进入了互联网行业，热衷于 Web 技术，同时对工业控制，ARM 嵌入式开发有一定的研究，喜欢旅行、电子游戏、运动以及摄影。其微博为@最亚伦。

孙辉，现为阿里巴巴的一只“前端攻城狮”，有多年的物联网及 Web 前端开发经验。爱美食、爱摄影，热爱一切富有创造性的事物。个人主页为 <http://www.sundway.me>。

目录

前言	XIX
1 Flux 是什么	1
Flux 是一套模式	1
数据入口	1
状态管理	2
保持同步更新	3
信息架构	4
Flux 并不是一个框架	4
Flux 的设计思路问题解决方案	5
数据流向	5
可回溯性	7
通知的一致性	8
简捷的架构分层	9
低耦合渲染	9
Flux 组件	10
动作	10
分发器	11
存储器	12
视图	12
安装 Flux 软件包	14
小结	16

2 Flux 的原则	17
MV*所面临的挑战	17
关注点分离	18
级联更新	19
模型更新的职责	20
单向数据	21
从开始到结束	22
无毒无害	23
显式优于隐式	23
暗藏隐患的更新	24
集中修改状态的地方	26
太多动作?	26
分层优于嵌套	27
多组件嵌套	27
嵌套深度与副作用	28
数据流和分层	28
应用数据和界面状态	29
两个相同的东西	29
强耦合转换	30
功能中心化	31
小结	31
3 搭建骨架架构	32
总体组织	32
目录结构	33
依赖管理	33
信息设计	34
用户不需要了解模型	34
存储器映射用户看到的内容	35
和哪些东西协同工作	36
在动作中注入存储器	36
获取 API 数据	36

改变 API 资源状态	42
本地动作	47
存储器和功能域	50
梳理顶层功能	50
无关紧要的 API 数据	51
构造存储器数据	53
模拟视图	53
找寻失去的数据	53
定位动作	54
端到端场景	56
动作清单	56
存储器清单	56
视图清单	57
小结	57
4 创建动作	58
动作的名称和常量	58
动作命名约定	58
静态动作数据	59
组织动作常量	62
特性动作生成器	63
什么时候需要模块化	63
模块化架构	64
模拟数据	65
模拟已存在的接口	65
模拟新接口	66
替换动作生成器	70
状态动作生成器	71
整合其他系统	72
web socket 连接	73
参数化动作生成器	76
删除多余的动作	76

保持动作的通用性	77
创建衍生动作	80
小结	81
5 异步动作	83
保持 Flux 同步	83
为什么要同步	83
压缩异步行为	84
异步动作语义	85
创建 API 调用	87
API 是常见的情况	87
API 调用和用户交互	88
结合 API 调用	92
复杂的动作生成器	93
组合动作生成器	96
返回 promise	97
不含 promise 的同步	98
组织异步行为	99
错误处理	101
小结	103
6 改变 Flux 存储器的状态	105
适应不断变化的信息	105
变化的 API 数据	105
变化的功能	106
受影响的组件	107
减少重复的存储器数据	107
通用存储器数据	107
注册通用存储器	108
结合通用和专用数据	112
处理存储器的依赖关系	116
等待存储器	116

数据依赖	118
UI 依赖	119
视图的更新顺序	125
存储器的注册顺序	125
视图渲染的优先级排序	125
处理存储器复杂度	126
存储器太多	126
反思功能域	126
小结	127
7 视图信息	128
传递视图数据	128
change 事件中的数据	128
视图决定何时渲染	132
保持视图无状态	135
UI 状态属于存储器	135
不用查询 DOM	135
视图的职责	136
渲染存储器数据	136
子视图结构	137
用户交互	138
在 Flux 中使用 ReactJS	138
设置视图状态	139
组成视图	143
响应事件	146
路由和动作	149
小结	154
8 信息的生命周期	155
组件生命周期难题	155
回收不再使用的资源	156
隐藏依赖	157

内存泄漏	157
Flux 结构是静态的	158
单例模式	158
与模型进行比较	161
静态视图	161
扩展信息	165
如何很好地扩展	165
最小化所需信息	169
扩展的动作	169
闲置的存储器	170
删除存储器数据	170
优化闲置的存储器	173
保持存储器数据	174
小结	182
9 不可变的存储器	183
放弃隐藏的更新	183
如何破坏 Flux 架构	184
获取存储器数据	186
一切皆不可变	187
强制执行单向数据流	187
纵横交错的单向数据流	188
过多的存储器?	189
没有足够的动作	189
强制不可变性	190
不可变数据的成本	195
垃圾回收是昂贵的	196
批量转换	196
抵消成本	197
使用 Immutable.js	197
不可变列表和映射	198
不可变的转换	201

变化检测	204
小结	208
10 实现分发器	209
抽象分发器接口	209
存储器的注册地	209
分发负载	210
依赖关系的处理	211
分发器所面临的挑战	212
教育的目的	212
单例模式的分发器	212
手动注册存储器	213
容易出错的依赖管理	213
构建分发器模块	214
封装存储器的引用	214
处理依赖	215
分发动作	217
优化存储器的注册	220
基础存储器类	221
一个动作方法	222
小结	226
11 可替代的视图组件	227
ReactJS 是适合 Flux 的	227
ReactJS 是单向的	227
重新渲染数据很简单	229
短小精悍的代码	229
ReactJS 的缺点	230
虚拟 DOM 和内存	230
JSX 和标记语言	231
库锁定	232
使用 jQuery 和 Handlebars	232

为什么是 jQuery 和 Handlebars	232
渲染模板	233
组合视图	236
事件处理	238
使用 VanillaJS	244
对可选择性保持开放	244
迁移到 React	244
新的技术热点	245
小结	245
12 使用 Flux 库	247
实现核心 Flux 组件	247
自定义分发器	247
实现一个基本的存储器	248
创建动作	248
实现中遇到的痛点	249
分发异步动作	249
划分存储器	249
使用 Alt	250
核心理念	250
创建存储器	251
声明动作生成器	253
监听状态变化	254
视图渲染以及分发动作	255
使用 Redux	258
核心思想	258
状态转换器和存储器	259
Redux 动作	262
渲染组件和分发动作	264
小结	269