O'REILLY®

TURING 图灵程序设计丛书

PWA开发实战

Building Progressive Web Apps

通过实践详解如何构建现代渐进式Web应用

[以] 塔勒·爱特尔 著

张俊达 译

中国工信出版集团

人民邮电出版社
POSTS & TELECOM PRESS

TURING 图灵程序设计丛书

PWA开发实战

Building Progressive Web Apps

[以] 塔勒·爱特尔 著
张俊达 译

Beijing • Boston • Farnham • Sebastopol • Tokyo O'REILLY®

O'Reilly Media, Inc.授权人民邮电出版社出版

人民邮电出版社
北 京

图书在版编目（CIP）数据

PWA开发实战 / （以）塔勒·爱特尔（Tal Ater）著 ；张俊达译. -- 北京 ： 人民邮电出版社，2019.1
（图灵程序设计丛书）
ISBN 978-7-115-50200-1

Ⅰ. ①P… Ⅱ. ①塔… ②张… Ⅲ. ①网页制作工具
Ⅳ. ①TP393.092.2

中国版本图书馆CIP数据核字(2018)第265309号

内 容 提 要

本书通过实际操作帮助读者透彻地理解现代渐进式 Web 应用开发，指导读者学会利用原生应用的特性构建 Web 应用。主要内容包括：某酒店网站构建全流程，开发渐进式 Web 应用时一些需要重点考虑的因素，离线优先的 Web 应用的原则，渐进式 Web 应用为用户界面带来的一些新挑战和新机会，等等。

本书适合 Web 开发人员和业务管理人员阅读。

◆ 著　　[以] 塔勒·爱特尔
译　　张俊达
责任编辑　岳新欣
责任印制　周昇亮

◆ 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
三河市君旺印务有限公司印刷

◆ 开本：800×1000　1/16
印张：13.75
字数：325千字　　2019年1月第1版
印数：1-3 000册　　2019年1月河北第1次印刷

著作权合同登记号　图字：01-2018-7358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：京东工商广登字 20170147 号

安徽大学图书馆 藏书 lib.ahu.edu.cn

献给我最爱的两位女士和她们的度假帽。

前言

渐进式 Web 应用（progressive Web app，PWA）是现代 Web 应用的一种激动人心的新形式。它利用了最新的 Web 功能，结合了原生移动应用的独特特性与 Web 的优点，为用户带来了新的体验。

本书将帮助你通过动手实践透彻地理解现代渐进式 Web 应用开发。

你将学习如何利用曾经专属于原生应用的特性构建 Web 应用。你将能够通过推送通知联系用户，在用户的主屏幕上占领先机，显著地加快网站速度，并且无论用户的网络连接状况如何，都能为其提供一个全功能的应用。

本书采用动手实践的学习方式，将一个现有网站逐章转化成现代渐进式 Web 应用。

读者对象

本书主要是为开发人员准备的。如果你想利用已有的 Web 开发技能，学习如何构建现代渐进式 Web 应用，那么本书就是为你而写的。

本书假定你至少对使用 HTML 和 JavaScript 进行 Web 开发有基本的了解，但不要求你熟悉 JavaScript 相对较新的特性，比如 ECMAScript 2015、promise，以及 ECMAScript 2017 的异步函数。如果你已经了解这些现代语言结构，完全可以跳过或快速浏览讲解这些内容的注释。

本书可以帮助非技术人员熟悉并基本了解现代渐进式 Web 应用的功能。其中很多章都包含了案例研究，这些案例是通过对世界上最有影响力的一些网站背后的团队进行访谈得到的，包括 Twitter、《华盛顿邮报》、Housing.com 和 Lyft。无论你是管理人员、设计师、产品经理，还是任何参与原生或 Web 应用决策的其他人员，了解如今可能实现的技术，将有助于你更有效地开展工作。

本书内容

哥谭帝国酒店是本书虚构的酒店，它的网站很简单。阅读本书时，你将接管这个网站，通

过 service worker 技术对它进行增强，使其几乎可以瞬间加载（哪怕是在最慢的连接状况下），并确保所有的功能都可以在用户完全离线时使用（包括查看用户的预订，甚至是发起新预订）。你将学习如何让用户添加一个图标到手机的主屏幕上，并从此处启动你的渐进式 Web 应用。最后，为了实现原生应用般的体验，你将添加推送通知，如此一来，就算用户离开你的网站，你也能联系并重新召回他们。

本书还探讨了开发渐进式 Web 应用时需要重点考虑的一些因素，并专注于帮助你切实理解这些概念，进而成为更高效的开发人员。此外，本书还介绍了实用的开发工具、安全因素，以及 service worker 的生命周期。

虽然本书大部分章节都集中在通过动手实践进行学习上，但其中两章（第 5 章和第 11 章）会让你思考渐进式 Web 应用提供的新功能，让你意识到它们不仅仅是应用于你的应用的一套新技巧。

第 5 章探讨了离线优先的 Web 应用原则，以这种方式构建的现代 Web 应用不会把失去网络连接视为错误，而是能够为之做好计划并优雅地处理。

第 11 章探讨了渐进式 Web 应用为用户界面带来的一些新挑战和新机会。渐进式 Web 应用改变了用户对 Web 应用的期望。其中一些挑战包括：增强用户的信任，让他们相信离线时自己的数据不会丢失；告诉用户离线时看到的内容可能是几小时之前的，并且让用户知道无论发生了什么重要的变化，应用会发送通知给他。若应对得当，这些挑战也是大好时机，可以借此增强用户对应用的信任，提高转化率，并在用户的手机上取得永久性位置。

本书最后介绍了一些即将到来的技术和浏览器 API，它们将进一步推动渐进式 Web 应用的发展。

排版约定

本书采用以下排版约定。

- **黑体**
  表示新术语或者重点强调的内容。
- 等宽字体（`constant width`）
  表示程序片段，以及正文中出现的变量、函数名、数据库、数据类型、环境变量、语句和关键字等。
- 等宽粗体（**`constant width bold`**）
  表示应该由用户输入的命令或其他文本。
- 等宽斜体（*`constant width italic`*）
  表示应该由用户替换或取决于上下文的值。

该图标表示渐进式 Web 应用的案例研究。

此为试读，需要完整PDF请访问：www.ertongbook.com

该图标表示一般说明。

该图标表示从另一个角度看待同一问题。

该图标表示警告或提醒。

代码示例

补充材料（包括示例代码、练习题等）可以从 https://github.com/TalAter/gotham_imperial_hotel 下载。

本书旨在帮助你做好工作。一般来说，你可以在程序和文档中使用本书的代码。除非你使用了很大一部分代码，否则无须联系我们获得许可。例如，使用本书的几段代码编写一个程序不需要获得许可，销售和分发 O'Reilly 书中示例的光盘则需要获得许可。引用书中示例代码来回答问题无须获得许可，把书中的大量示例代码放入你的产品文档则需要获得许可。

我们很希望但并不强制要求你在引用本书内容时加上引用说明。引用说明一般包括书名、作者、出版社和 ISBN。例如："*Building Progressive Web Apps* by Tal Ater (O'Reilly). Copyright 2017 Tal Ater, 978-149-196165-0"。

如果你觉得自己对示例代码的使用超出了合理引用或者上述许可的范围，请随时通过 permissions@oreilly.com 联系我们。

O'Reilly Safari

Safari（之前称作 Safari Books Online）一个针对企业、政府、教育者和个人的会员制培训和参考平台。

会员可以访问来自 250 多家出版商的上千种图书、培训视频、学习路径、互动式教程和精选播放列表，这些出版商包括 O'Reilly Media、Harvard Business Review、Prentice Hall Professional、Addison-Wesley Professional、Microsoft Press、Sams、Que、Peachpit Press、Adobe、Focal Press、Cisco Press、John Wiley & Sons、Syngress、Morgan Kaufmann、IBM Redbooks、Packt、Adobe Press、FT Press、Apress、Manning、New Riders、McGraw-Hill、Jones & Bartlett、Course Technology 等。

要了解更多信息，可以访问 http://www.oreilly.com/safari。

联系我们

请把对本书的评价和问题发给出版社。

美国：

O'Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472

中国：

北京市西城区西直门南大街 2 号成铭大厦 C 座 807 室（100035）
奥莱利技术咨询（北京）有限公司

O'Reilly 的每一本书都有专属页面，你可以在那里找到本书的相关信息，包括勘误表、示例以及其他信息。[1] 本书的网页地址是：http://shop.oreilly.com/product/0636920052067.do

对于本书的评论和技术性问题，请发送电子邮件到：bookquestions@oreilly.com

要了解更多 O'Reilly 图书、培训课程、会议和新闻的信息，请访问以下网站：http://www.oreilly.com

我们在 Facebook 的地址如下：http://facebook.com/oreilly

请关注我们的 Twitter 动态：http://twitter.com/oreillymedia

我们的 YouTube 视频地址如下：http://www.youtube.com/oreillymedia

致谢

我首先要感谢 Alex Russell 和 Jake Archibald。在 2014 年的谷歌 I/O 开发者大会上，我偶然听到了他们的演讲。在那之前，我甚至不知道本书中涵盖的任何内容意味着什么。他们的演讲和本书的写作帮助我增加了了解。

我还要感谢在现实世界中构建渐进式 Web 应用的优秀人才，特别是那些在本书中分享经验的人。感谢 Joey Marburger、Ritesh Kumar、Rahul Yadav、Nicolas Gallagher、Chris Nguyen 和 Jeremy Toeman。

感谢 O'Reilly 团队帮助我出版了本书，包括 Ally MacDonald、Jeff Bleiel、Sonia Saruba、Colleen Cole、David Futato、Rebecca Demarest、Heather Scherer、Ellen Troutman、Amanda Kersey，以及 Karen Montgomery 和她的戴胜鸟（本书封面上的动物）。

还要感谢谷歌、Opera、Mozilla 和微软的开发团队，在他们的帮助下，本书和本书中介绍的技术才成为了现实。感谢优秀的开发者们帮助我更好地理解了这项技术。感谢 Jeffrey Posnick、Addy Osmani、Matt Gaunt 和 Paul Kinlan。

注 1：本书中文版的勘误请到 http://www.ituring.com.cn/book/2040 查看和提交。——编者注

感谢 Alex Feyerke 和 Hoodie 团队，他们围绕离线优先做出了开创性的工作，并通过写作记录下来。本书中的不少内容都是受到了他们的启发。

编写这样一本书最困难的一点，是在一年多的时间里几乎没有收到外部反馈，这带来了很大的不确定性。我要感谢所有在本书写作过程中抽出时间向我提供反馈意见的人。感谢 James Stanley、Patrick Conant、Fabio Rotondo、Neville Franks 和 Florian Semrau。

最后，我要感谢本书的技术审阅者，他们对本书进行了全面的技术审查，以保障我所写的内容确实有意义。感谢 Andreas Bovens、Kenneth Rohde Christiansen、Patrick Kettner 和 Thomas Steiner。

电子书

扫描如下二维码，即可购买本书电子版。

目录

前言 xi

第 1 章　渐进式 Web 应用介绍 1

1.1　Web 反击战 2
1.2　当前的移动领域 2
1.3　渐进式 Web 应用的优势 4
1.4　浏览器标签页、Web 和 service worker 6

第 2 章　你的第一个 service worker 8

2.1　设置示例项目 8
2.2　欢迎来到哥谭帝国酒店 9
2.3　熟悉代码 10
2.4　当前的离线体验 11
2.5　创建你的第一个 service worker 13
2.6　什么是渐进增强 16
2.7　HTTPS 和 service worker 16
2.8　从 Web 获取内容 17
2.9　捕获离线请求 18
2.10　创建 HTML 响应 19
2.11　理解 service worker 作用域 21
2.12　小结 22

第 3 章　CacheStorage API 23

3.1　CacheStorage 是什么，不是什么 24
3.2　决定何时进行缓存 24
3.3　在 CacheStorage 中存储请求 25
3.4　从 CacheStorage 中取回请求 26

3.5 在示例应用缓存……27
3.6 匹配每个请求的正确响应……29
3.7 HTTP 缓存和 HTTP 头……31
3.8 小结……31

第 4 章 service worker 生命周期和缓存管理……33

4.1 service worker 生命周期……36
4.2 service worker 的生命周期与 `waitUntil` 的重要性……38
4.3 更新 service worker……39
4.4 为什么需要管理缓存……40
4.5 缓存管理与清除旧缓存……42
4.6 重用已缓存的响应……46
4.7 配置服务器以提供正确的响应头部……47
4.8 开发者工具……48
4.8.1 控制台……48
4.8.2 清除缓存并刷新……48
4.8.3 检查 CacheStorage 和 IndexedDB……49
4.8.4 网络节流与模拟离线情况……49
4.8.5 Lighthouse……50
4.9 小结……50

第 5 章 拥抱离线优先……51

5.1 什么是离线优先……52
5.2 常用缓存模式……53
5.3 混合与匹配：创造新模式……55
5.4 规划缓存策略……57
5.5 实现缓存策略……59
5.6 App shell 架构……68
5.7 实现 App shell……70
5.8 解锁成就……72
5.9 小结……73

第 6 章 使用 IndexedDB 在本地存储数据……74

6.1 什么是 IndexedDB……75
6.2 使用 IndexedDB……77
6.2.1 打开数据库连接……77
6.2.2 数据库版本 / 修改对象存储……78
6.2.3 添加数据到对象存储……79
6.2.4 从对象存储中读取数据……80
6.2.5 IndexedDB 版本管理……81
6.2.6 使用游标读取对象……82
6.2.7 创建索引……84
6.2.8 使用索引读取数据……85

6.2.9 限制游标的范围 ····· 86
6.2.10 设置游标方向 ····· 87
6.2.11 更新对象存储中的对象 ····· 87
6.2.12 从对象存储删除对象 ····· 88
6.2.13 从对象存储中删除所有对象 ····· 89
6.2.14 处理冒泡 IndexedDB 错误 ····· 89
6.3 SQL 忍者的 IndexedDB ····· 90
6.4 IndexedDB 实践 ····· 91
6.5 promise 式的数据库 ····· 98
6.6 IndexedDB 管理 ····· 103
6.7 在 service worker 中使用 IndexedDB ····· 104
6.8 IndexedDB 生态系统 ····· 105
6.8.1 PouchDB ····· 105
6.8.2 localForage ····· 106
6.8.3 Dexie.js ····· 106
6.8.4 IndexedDB Promised ····· 107
6.9 小结 ····· 107

第 7 章 使用后台同步保证离线功能 ····· 108

7.1 后台同步是如何工作的 ····· 109
7.2 SyncManager ····· 111
7.2.1 访问 SyncManager ····· 111
7.2.2 注册事件 ····· 112
7.2.3 sync 事件 ····· 112
7.2.4 事件标签 ····· 112
7.2.5 获取已注册 sync 事件列表 ····· 113
7.2.6 最后的机会 ····· 113
7.3 传递数据给 sync 事件 ····· 114
7.3.1 在 IndexedDB 中维护操作队列 ····· 114
7.3.2 在 IndexedDB 中维护请求队列 ····· 116
7.3.3 传递数据给 sync 事件标签 ····· 118
7.4 给应用添加后台同步 ····· 118
7.5 小结 ····· 125

第 8 章 使用 `postMessage()` 在 service worker 和页面之间通信 ····· 126

8.1 窗口向 service worker 通信 ····· 127
8.2 service worker 向所有打开的窗口通信 ····· 128
8.3 service worker 向特定窗口通信 ····· 130
8.4 使用 MessageChannel 保持通信渠道打开 ····· 131
8.5 窗口间的通信 ····· 133
8.6 从 sync 事件向页面传递消息 ····· 136
8.7 小结 ····· 137

此为试读，需要完整PDF请访问：www.ertongbook.com

第 9 章　可安装的 Web 应用：占领主屏先机 ····· 138

9.1　可安装的 Web 应用 ····· 139
9.2　浏览器如何决定何时显示应用安装横条 ····· 140
9.3　剖析 Web 应用清单 ····· 141
9.4　各端兼容性 ····· 145
9.5　小结 ····· 146

第 10 章　推送通知 ····· 147

10.1　推送通知的生命周期 ····· 147
10.1.1　Notification API ····· 147
10.1.2　Push API ····· 148
10.1.3　Push+Notification ····· 150
10.2　创建通知 ····· 150
10.2.1　请求通知权限 ····· 150
10.2.2　显示通知 ····· 153
10.2.3　为哥谭帝国酒店添加通知支持 ····· 157
10.3　为用户订阅推送事件 ····· 158
10.3.1　生成 VAPID 公钥和私钥 ····· 160
10.3.2　生成 GCM 密钥 ····· 161
10.3.3　创建新订阅 ····· 162
10.3.4　为哥谭帝国酒店用户订阅推送消息 ····· 164
10.4　从服务端发送推送事件 ····· 166
10.5　监听推送事件并显示通知 ····· 168
10.6　小结 ····· 174

第 11 章　渐进式 Web 应用的用户体验 ····· 175

11.1　优雅与信任 ····· 175
11.2　从 service worker 传递状态 ····· 176
11.3　使用 Progressive UI KITT 通信 ····· 178
11.4　渐进式 Web 应用中的常见消息 ····· 180
11.4.1　缓存完成 ····· 180
11.4.2　页面已缓存 ····· 180
11.4.3　操作失败，但会在用户恢复连接时完成 ····· 181
11.4.4　启用通知 ····· 181
11.5　选择正确的用词 ····· 181
11.6　不要直奔主题 ····· 182
11.7　渐进式 Web 应用的设计 ····· 184
11.7.1　设计应该反映条件的变化 ····· 184
11.7.2　设计应该适应运行环境 ····· 185
11.7.3　设计应该适应每种媒介的特殊性 ····· 185
11.7.4　设计应该向用户注入信心并通知用户 ····· 186
11.7.5　设计应该帮助用户和企业实现目标 ····· 186

11.8 负责安装提示……186
11.9 使用 RAIL 测量性能并实现高性能……187
11.10 小结……189

第 12 章 渐进式 Web 应用的未来……190

12.1 使用 Payment Request API 接受支付请求……190
12.2 使用 Credential Management API 进行用户管理……192
12.3 WebGL 实时图像处理……193
12.4 未来的语音识别 API……194
12.5 使用 WebVR 在浏览器中实现虚拟现实……194
12.6 轻松共享应用……195
12.7 流畅的媒体播放 UI……196
12.8 下一个伟大时代……197

附录 A service worker：采用 ES2015 的大好时机……198

附录 B 全页间隙式广告……201

附录 C CORS 与 NO-CORS……202

关于作者……204

关于封面……204

第1章

渐进式Web应用介绍

> 词是被遗忘的名之浅影。因为名有力量，所以词也有力量。词可以点燃人们心中的火焰，也可以让最狠的心流下眼泪。七个词便可以让一个人爱上你，十个词便可以让最坚强的人顿失意志。但是词不过是火之描绘，而名才是火之本身。
>
> ——Patrick Rothfuss，《风之名》

每隔几年，Web 就会经历一个关键性时刻。在这个时刻，几种独立的技术相互碰撞，在公众中引起轰动。这些技术可能已经诞生多年，也可能是刚刚获得浏览器支持的新技术。但对于旁观者来说，似乎在一个闪光的瞬间，Web 突然向前跃进了一步。

Ajax 技术就是这样。它似乎是某一天不知从何处突然蹦出来的（尽管很多底层技术已经存在多年，如 XMLHttpRequest），改变了我们对 Web 的理解：一系列相互链接的、大部分是静态的页面。

Ajax 本身只是 Web 2.0 革命的一部分，而 Web 2.0 是另一个强大的名字，它在 2004 年从天而降，一夜之间引爆了世界。

几年后，**移动优先**（mobile-first）出现了，它标志着我们对 Web 开发的看法发生了转变。通过给一套设计原则命名，这两个单词获得了难以置信的力量。只用这两个单词，我们就可以大声说，用户坐在台式机前，用 20 英寸的显示器和一根连接到墙上的电缆上网的时代已经结束了。这两个单词让我们明白，是时候改变 Web 开发的方式了。

这样的时刻往往不是在技术诞生的时候出现的，而是在它被命名的时候。

名字就是有这样的力量：它让我们掌握新思想、讨论新概念，提醒我们注意在表面下酝酿的风暴。

一个同样巨大的转变正在发生。幸运的是，它有一个名字。[1]

1.1 Web反击战

渐进式 Web 应用是一种崭新的 Web 应用，它结合了原生应用的优点和 Web 少冲突的特点。

渐进式 Web 应用始于简单的网站，但随着用户的使用，它不断获得新的权限，并从网站变成一种更像传统原生应用的形式。

想象一下，你早上醒来，抓起手机，浏览本地列车公司的网站。你快速查看了上班需要乘坐的列车的时间表，然后关闭浏览器，并把手机放进口袋。下班时，你再次访问该网站，查看下一趟列车何时发车（你甚至没有注意到正在乘坐的电梯没有手机信号，因为列车公司的网站依然在运行，即使你处于离线状态）。次日，当你再次访问该网站时，浏览器询问你是否要添加快捷方式到主屏幕上，你欣然同意。当天晚些时候，当你从主屏幕的图标登录该网站时，它通知你，由于施工，列车可能会延误，并问你是否想接收关于列车时刻变化的后续通知。第三天早上，当你醒来时，手机收到消息推送，说该列车会延误 15 分钟。你按下闹钟上的延时按钮，又多睡了一会。

渐进式 Web 应用从一个简单的网站开始，慢慢获得新的权限，直到和原生应用一样，而不是试图把你送到应用商店，希望你安装应用。就这样，列车公司在你的手机上一步一步赢得了永久性的位置。

这种新的渐进增强模型，取代了只提供“安装”和“不安装”两种选择的原生应用。渐进式 Web 应用能赢得用户的信任，并按需获得新的权限。

你可能会问，为什么这是对原生应用的改进呢？我们为什么不坚持使用原生应用呢？好吧，除非你是少数的幸运儿，否则你应该知道，原生应用已经行不通了。用户安装应用的可能性在逐年减小，获取新用户的成本在逐渐增长，留住用户变得越来越困难。

1.2 当前的移动领域

当第一款 iPhone 在 2007 年推出时，它的杀手级功能是让你能够在手机上浏览网站。当移动应用于一年后诞生时，开发人员终于能够突破网页的功能限制了（同时，由于应用商店的引入，也面临许多新的限制）。

Web 具有高级图像技术、地理位置识别、消息推送、离线可用性、主屏幕图标等特性，但在许多开发人员的眼中，Web 似乎相形见绌。原生应用接管了全球（和我们的手机）。

但这种趋势正在转变。虽然我们在手机和移动应用上花费的时间比以往任何时候都多，但我们使用的应用却越来越少。用户安装的应用少了，而且只使用其中几个。如果你的应用在应用商店中排在前 10 位，你可能不会有这种困扰。但是，现在尝试将一款新的应用打

注 1：更确切地说，幸运的是，Alex Russel 和 Frances Berriman 有一天共进晚餐，并想出了一个名字。详细内容请参见 Alex Russel 的博客文章：“Progressive Web Apps: Escaping Tabs Without Losing Our Soul”。

入市场几乎是不可能的，成本就更别提了。

移动用户的行为

根据 2016 年 comScore 的报告，在移动设备上，平均每人将 84% 的时间花在最流行的 5 个应用上。抱歉，这些不是你的应用。在平板电脑上，这个比例甚至更高，用户将 95% 的时间花在了这 5 个应用上。

该报告还表明，移动网站比原生应用更容易获得大量用户。拥有 500 万以上访问者的移动网站接近 600 个，比拥有类似受众的原生应用多 4.5 倍。排名前 1000 位的移动网站拥有的用户数量几乎是排名前 1000 位的原生应用的 3 倍，而且前者的用户增长速度是后者的两倍。

让用户安装并使用应用，如同在一个漏斗形空间里挣扎求生。用户需要知道你的应用（通过传统的在线广告或你的网站），然后必须访问其在应用商店中的页面，接下来需要点击安装。他们需要同意授予应用不同的权限，然后等待应用下载并安装。最后，他们至少要启动一次应用，甚至使用它。

当用户安装他们知道并喜欢的应用（比如 Twitter 或者 Facebook）时，这个漏斗看起来并不算太糟糕。但是大量研究表明，在这个漏斗的每一个环节，平均有 20% 的用户丢失了。应用开发人员为广告点击付费后，发现只有不到 20% 的用户实际启动了应用，这种情况并不罕见。

网站竭尽所能地让你安装他们的应用，甚至采用了一种新的广告方式。你肯定见过这种广告：当你打开一个网站，想看一篇短文或者查询明天的天气时，发现所需信息近在眼前，但是一个横幅广告随即弹了出来，挡住了你想看的内容。它问你是否愿意安装一个应用，而不直接阅读已经在你眼前的内容。

有些人称之为**全页间隙式广告**（full-page interstitial ad）。我喜欢一个较短的名字：**用力关门**（the door slam，如图 1-1 所示）。要详细了解全页间隙式广告的作用与副作用，请参见附录 B。

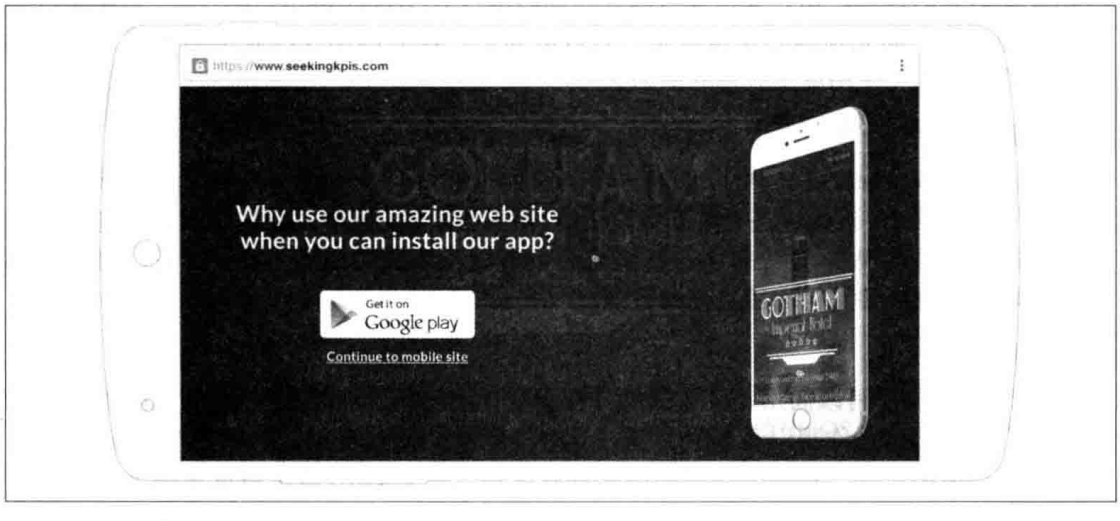

图 1-1：一种常见的“用力关门”式广告

让用户在手机上安装你的应用是一场残酷、昂贵的战斗。然而，原生应用相较于 Web 的优势，让应用开发者心甘情愿为每一次应用安装忍受痛苦。

与传统网站不同，原生应用的生命并不限于用户首次发现它到离开这段短暂的时间（有时候，这段时间只有几秒钟）。一旦安装，原生应用就在你的主屏幕上占据了永久性的位置。它可以随时通过消息推送提醒你它的存在。这让开发者有很多机会在应用的长生命周期里尝试获得投资回报。

但随着渐进式 Web 应用的推出，潮流终于开始转向了。这些超能力所带来的优势曾经是原生应用专有的，但现在可以移植到 Web 应用上了。将这些能力与 Web 少冲突、一步漏斗的特点（使用点击链接取代安装应用）结合起来，你就会明白为什么用户、Web 开发者和企业都能因拥抱渐进式 Web 应用而获益良多。

1.3 渐进式Web应用的优势

随着上述超能力的引入，渐进式 Web 应用实现了我们寄予原生应用的很多期望。

以下是本书中将介绍的部分优势。

无连接状态下的可用性

和传统网站不同，渐进式 Web 应用不依赖于用户的连接状态。当用户访问一个渐进式 Web 应用时，它会注册一个 service worker（参见 1.4 节），service worker 可以检测并响应用户连接状态的变化。无论用户是离线、在线，还是处于网络不稳定状态，它都可以提供完整的用户体验。

用户可以在穿越大西洋的航班上使用你的渐进式 Web 应用，甚至可以进行操作（例如发布信息、回复事件或者评论文章），用户知道他的操作会在网络恢复之后立刻完成，即便他关闭了应用和浏览器。详情请参见第 7 章。

渐进式 Web 应用引入了一定程度的可靠性，将用户的信任程度提升到原本只有原生应用才能达到的程度。用户知道他可以在任何时间打开 WhatsApp 应用，写一条短信，然后关掉手机，而无须担心连接状态。到目前为止，Web 网站还没获得这种信任，这也是用户更偏向于使用原生应用的原因之一。

加载速度快

使用 service worker，我们可以创建一个瞬间运行的网站，无论用户的网速极快，还是使用不可靠的 2G 连接，甚至是在完全没有网络连接的状态下。网站可以在几毫秒内加载，这比过去的 Web 要快得多，甚至比原生应用还要快。在第 5 章中，我们将学习如何实现这一点，并探讨离线优先的原则。

推送通知

渐进式 Web 应用可以向用户推送通知（甚至是在用户离开网站几天后）。这些通知提供了一个好机会，让你得以重新吸引用户，并提醒他们回到应用。渐进式 Web 应用的通知是完全原生化的，和原生应用的通知没有区别。参见第 10 章了解更多关于推送通知的内容。

此为试读，需要完整PDF请访问：www.ertongbook.com