

扩展及应用开发

Chrome

李喆 编著

- 使用扩展装备你的浏览器
- 借助HTML5创建桌面应用



人民邮电出版社
POSTS & TELECOM PRESS

Chrome

扩展及应用开发

李喆 编著



人民邮电出版社
北京

图书在版编目 (CIP) 数据

Chrome 扩展及应用开发 / 李喆编著. -- 北京: 人民邮电出版社, 2014.10

(图灵原创)

ISBN 978-7-115-36866-9

I. ① C… II. ① 李… III. ① 浏览器—程序设计
IV. ① TP393.092

中国版本图书馆 CIP 数据核字 (2014) 第 199719 号

内 容 提 要

本书简述了 Chrome 平台开发的概念、扩展与应用的区别及其常用功能,并结合具体场景给出了实例。主要介绍了 Chrome 扩展基础、Chrome 扩展的 UI 界面、管理浏览器和部分高级 API 划分,讲解了文件系统、网络通信、操作 USB 设备和串口通信等多个与操作系统紧密相关的内容,还介绍了 Chrome 主题开发、支持国际化的程序开发,以及如何使用 Angular 框架进行开发。

本书以直观生动的实例帮助读者理解知识要点,以自顶向下的方式进行讲解,让读者先接触最直观的内容,然后慢慢深挖究其本质,非常适合开发者自学。

◆ 编 著 李 喆
责任编辑 王军花
执行编辑 张 霞 李鸿鹏
责任印制 焦志炜
封面创意 李 喆
封面设计 出壳创意

◆ 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: 15.75

字数: 372 千字

2014 年 10 月第 1 版

印数: 1~3000 册

2014 年 10 月北京第 1 次印刷



定价: 49.00 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

推荐序

Google Chrome 是 21 世纪以来最伟大的浏览器之一。

对 Google 而言，它是称雄 Web 时代的重要基石。

对开发者而言，它史无前例地友好。

对用户来说，它拥有超快的速度与优秀的使用体验。

基于优秀的可编程架构，成百上千的开发者贡献了数以万计的扩展和应用。和“前辈”Firefox 相比，Chrome 已逐渐超出浏览器边界，成为不容忽视的应用平台。

如今，用户可以一边浏览网页，一边听音乐，同时还可以玩游戏，也可以通过 Chrome App 离线编辑文档，处理图形图像。

大型商业公司（如天猫）也开发了大量的 Chrome 扩展，用于检测质量和性能，辅助开发，提升数据效率等。

本书系统化地介绍了 Chrome 平台的扩展构架和机制，通过实际可执行的代码由浅入深地介绍了如何完整地开发 Chrome 扩展。通过阅读本书，读者对 W3C 标准将会有更细致的理解，并且可以“零距离”地观摩大型技术平台对规范与标准的实现。

对正在或有志于从事互联网行业的开发者而言，这将是一个非常良好的学习体验。

第一次认识 Sneezry 是在图灵社区 (iTuring.cn)，进一步交流后才了解到他是一位在校学生。而且最初本书是以电子书的形式在图灵社区公开发布的，供读者免费阅读，广受好评，后来经过进一步的修改与完善，才有了现在这本书。作为技术人，我深知撰写专业书实属不易，尤其这种对技术的无私精神值得所有人喝彩。

很高兴看到越来越多的学生参与开源世界，以自己的方式为之贡献力量。

天猫前端技术专家舒文

2014.08.14

感谢我的父母和家人，感谢他们支持我写完这本书，谨以此书献给我的妈妈、爸爸和闹闹。对了，闹闹是一只棕红色的阿拉斯加犬，它是我的好朋友，也是我们家庭中的一员。

前言

一个电子专业的在校学生，每天学习的是电子在晶格中如何游走，研究的是半导体器件的电学特性，无论如何都不会与这本书的作者联系在一起。

说起来，写这本书非常偶然。某一天我突然就想写点什么了，想写点很多人都会看的东西，作为人生中的一个成就。虽然我的专业是电子，但编程一直都是我最大的爱好，前端更是我最熟悉的领域。作为 Google 的追随者，我是第一批使用 Chrome 浏览器的用户，也是 Chrome 推出扩展功能后较早投入其中的开发者之一，所以 Chrome 开发自然就成为了我写作的首选。

我在 2010 年 9 月份开始接触 JavaScript，当时这门与 HTML 和 CSS 巧妙结合在一起的编程语言立刻就迷住了我。在刚刚接触 JavaScript 时，我常常写一些简单的代码段来更改网页的外观或增强浏览器功能，比如将当前网页背景更改成类似安卓手机的动态壁纸，或者是自动给出鼠标选中单词的翻译，等等。这些代码段被称作 bookmarklet（书签小程序），编写 bookmarklet 是我第一次通过 JavaScript 和浏览器打交道。接下来我编写了第一个 Google 在线输入法的 bookmarklet，甚至早于 Google 官方。2011 年 1 月 Google 输入法的工程师们通过这个小程序联系到了我，并鼓励我到 Google 实习。

随着我编写的 bookmarklet 功能越来越复杂，简单的代码段已经无法实现我创作的灵感，于是我开始试着开发 Chrome 扩展程序。Chrome 扩展程序不仅仅可以获取到更高的权限，实现更加复杂的功能，而且还可以让用户感觉这个小程序本身就是浏览器的一部分。在接下来的一年里，我发布到网上商店的扩展有 20 余款（由于网上商店有每个开发者最多发布 20 款扩展应用的限制，因而部分扩展后来被我下架了），而那些我为自己临时需求而编写的没有发布到网上商店的扩展则数不胜数，由此我意识到 Chrome 扩展已经真正融入到了我的生活。这些扩展中我认为最成功的是一款名为 Dualx 的即时通信程序，2013 年 4 月火狐的工程师通过 Dualx 联系到了我，并邀请我到火狐参与火狐浏览器扩展的开发工作。

如果说 Chrome 扩展让我重新认识了浏览器体验，那么 Chrome 应用可以说让我重新认识了软件开发。之前我用 VC 和 VB 编写过带有可视化窗口的程序，但是它们看起来总是很丑，而 HTML 和 CSS 却可以通过几行简单的代码编写出一个绚丽的页面。当 Google 提出 Packaged App 的概念时，我意识到利用 HTML 和 CSS 编写程序窗口将是一件多么令人兴奋的事情！经

过 Google 工程师认真优化过的 JavaScript 引擎确保了 JavaScript 可以胜任大型程序的逻辑语言，Google 对 Chrome OS 的强力推广确保了 Chrome 应用的市场地位，于是，越来越多的公司开始将产品延伸到 Chrome 应用这一新的平台上。

这本书诞生于图灵社区，图灵社区的写作氛围很好，而且在线编辑器支持 Markdown 语法。一开始我并不好意思直接说写的是书，直到定稿前夕我才把“文集”二字改成了“书”。回顾几个月之前，一拍脑门夹着笔记本就去图书馆开写了。在写作的过程中遇到一个又一个坑，有时为了让一个实例跑通，要调上一整个下午。如果当初我知道会遇到这么多困难，想来是不会动笔的。但既然动笔了，半途而废着实没有颜面，所以就一直坚持到了最后。

写作对个人能力的提升是非常大的。由于每一个知识点都必须准确，不可含糊其辞，所以我在写作的过程中不得不一遍遍仔细翻阅 Chrome 官方开发文档和 W3C 标准，同时还要编写实例进行验证。

值得庆幸的是，这本书还没有完成就得到了很多读者的关注，他们给了我很大的鼓励，有的读者还表示可以无私地帮我审校书稿。

在此我要重点感谢方觉。大家可能对这个名字并不熟悉，但他创建和维护的 crxdoc-zh.appspot.com，相信开发 Chrome 扩展和应用的人没有几个不知道的。这本教程也参考和引用了该网站的部分内容，在此表示感谢。他不仅帮助我仔细地推敲本书的语言表述，而且还纠正了一些知识错误，甚至是官方文档中同样出现的错误。这让我感到十分惊讶，后来才发现他还是 Chromium 项目的贡献者，就更加对他敬佩不已了。

吕鹏同学是很早就在互联网上结识的朋友，他和我一直在一起鼓捣些小程序。能找到志同道合的小伙伴非常不易，我十分庆幸能在广阔的互联网中与他结识。初识吕鹏时，他还是个大二的在校生，转眼就已是微软的大牛了，我为能有如此优秀的朋友感到自豪。吕鹏是第一个知道我要写这本书的人，在写作的前期他给了我莫大的鼓励，而这是促成本书的重要因素。本书初稿写成后，他也参与了审校，并在最终定稿前对本书的结构和配图给出了非常好的建议，在此非常感谢他所提供的帮助！

在我刚刚开始写独立博客时，李典同学就成为了我的读者，虽然他一直躲在 Google Reader 后面。李典同学是第一个为此书提交勘误表的读者。在早先他为我一个项目贡献代码时，我就发现他是一个做事十分认真的人，这次也不例外，以至于后来我将他的勘误表作为样板发给了参与审校的每一位读者。感谢李典同学为这本书作出的贡献！

参与审校的还有赵余和韩骏，在此一并表示感谢！

最后我还要感谢图灵公司的杨帆、张霞和傅志红老师的鼓励和建议，感谢李鸿鹏编辑和张霞编辑的细心修改，没有他们的帮助，这本书就无法与读者见面。

由于水平有限，书中不免出现错误，欢迎读者朋友指正。你可以通过 lizhe@lizhe.org 与我联系，也可以通过图灵社区在线提交勘误信息，在此先行感谢。

第 1 章	初步接触 Chrome 扩展及应用开发	001
1.1	认识 Chrome 扩展及应用	002
1.2	应用与扩展的区别	003
1.3	我的第一个 Chrome 扩展	004
1.4	Manifest 文件格式	008
1.5	DOM 简述	011
1.6	调试方法与代码质量	013
第 2 章	Chrome 扩展基础	016
2.1	操作用户正在浏览的页面	017
2.2	跨域请求	020
2.3	常驻后台	025
2.4	带选项页面的扩展	027
2.5	扩展页面间的通信	032
2.6	存储数据	034
2.7	il8n	037
第 3 章	Chrome 扩展的 UI 界面	039
3.1	CSS 简述	040
3.1.1	默认样式及 box 模型	040
3.1.2	元素定位	042
3.1.3	字体及背景颜色	043
3.2	Browser Actions	045
3.2.1	图标	045
3.2.2	popup 页面	047

3.2.3	标题和 badge	049
3.3	右键菜单	051
3.4	桌面提醒	056
3.5	omnibox	058
3.6	Page Actions	061
第 4 章	管理你的浏览器	062
4.1	书签	063
4.1.1	创建书签	064
4.1.2	创建书签分类	064
4.1.3	调整书签位置	065
4.1.4	更新书签	065
4.1.5	移除书签	065
4.1.6	获取书签内容	065
4.1.7	书签事件	066
4.2	Cookies	068
4.2.1	管理 Cookie	068
4.2.2	读取 Cookie	069
4.2.3	设置 Cookie	069
4.2.4	删除 Cookie	070
4.2.5	获取全部的 cookie store	070
4.2.6	监控操作 Cookie 的行为	071
4.3	历史	071
4.3.1	读取历史	072
4.3.2	添加历史	073
4.3.3	删除历史	074
4.3.4	对历史操作的监听	074
4.4	管理扩展与应用	075
4.4.1	读取用户已安装的扩展和应用的信息	075
4.4.2	获取权限警告	076
4.4.3	启用、禁用、卸载扩展和启动应用	076
4.4.4	对管理操作的监听	077
4.5	标签	078
4.5.1	获取标签信息	078
4.5.2	创建标签	079
4.5.3	更新标签	079

4.5.4	移动标签	080
4.5.5	重载标签	080
4.5.6	移除标签	080
4.5.7	获取当前标签页面的显示语言	081
4.5.8	获取指定窗口活动标签可见部分的截图	081
4.5.9	注入 JavaScript 和 CSS	081
4.5.10	与指定标签中的内容脚本 (content script) 通信	082
4.5.11	监视标签行为	082
4.6	Override Pages	084
第 5 章	部分高级 API	086
5.1	下载	087
5.2	网络请求	089
5.3	代理	092
5.4	系统信息	095
第 6 章	Chrome 应用基础	098
6.1	更加严格的内容安全策略	099
6.2	图标设计规范	101
6.3	应用的生命周期	103
6.4	应用窗口	106
6.4.1	创建窗口	106
6.4.2	样式更加自由的窗口	109
6.4.3	获取窗口	115
6.4.4	窗口事件	116
6.5	编写第一个 Chrome 应用	117
第 7 章	文件系统	125
7.1	目录及文件操作对象	126
7.2	获取目录及文件操作对象	127
7.3	读取文件	128
7.4	遍历目录	130
7.5	创建及删除目录和文件	132
7.6	写入文件	134
7.6.1	Typed Array	134
7.6.2	Blob 对象	136

7.6.3	FileWriter 对象	137
7.7	复制及移动目录和文件	139
第 8 章	媒体库	140
8.1	获取媒体库	141
8.2	添加及移除媒体库	146
8.3	更新媒体库	148
8.4	获取媒体文件信息	152
第 9 章	网络通信	154
9.1	UDP 协议	155
9.1.1	建立与关闭连接	156
9.1.2	发送与接收数据	158
9.1.3	多播	160
9.1.4	获取 socket 和组	162
9.1.5	局域网聊天应用	163
9.2	TCP 协议	166
9.2.1	建立与关闭连接	167
9.2.2	发送与接收数据	168
9.2.3	获取 socket	170
9.3	TCP Server	170
9.3.1	建立与关闭连接	171
9.3.2	监听数据	173
9.3.3	获取 socket	174
9.3.4	HTTP Server	175
9.4	WebSocket	178
第 10 章	其他接口	179
10.1	操作 USB 设备	180
10.1.1	发现设备	180
10.1.2	接口操作	181
10.1.3	传输操作	182
10.2	串口通信	183
10.2.1	建立连接	183
10.2.2	发送和接收数据	184
10.2.3	获取连接及状态	185

10.3 文字转语音	185
10.3.1 朗读文字	186
10.3.2 获取声音	187
10.3.3 获取朗读状态并监听事件	188
10.4 系统信息	189
 附录 A 制作 Chrome 主题	191
A.1 背景图片	193
A.2 颜色	198
A.3 颜色叠加	205
A.4 UI 属性	206
A.5 使用 My Chrome Theme 制作主题	209
 附录 B CSS 选择器	212
B.1 基本选择器	213
B.2 高级选择器	215
B.3 伪类	216
 附录 C 初识 AngularJS	220
C.1 视图	221
C.2 \$scope	224
C.3 module 与路由	225
 附录 D Chrome 扩展及应用的完整 API 列表	229
D.1 Chrome 扩展的全部 API	230
D.2 Chrome 应用的全部 API	235

第1章

初步接触Chrome 扩展及应用开发



本章首先会对 Chrome 扩展应用进行简单的概述，然后将带着读者编写一个简单的扩展，使读者对扩展的认识更加深入。在讲解扩展 Manifest 文件格式时，也会简单讲解一下 JSON（JavaScript Object Notation）这一轻量级的数据交换格式，从而避免没有接触过 JSON 的读者在阅读后续内容时遇到困难。另外，本章还会简单讲解一下 DOM，这对从未接触过网页编程的读者会非常有帮助。

1.1 认识 Chrome 扩展及应用

Chrome 是 Google 公司基于 WebKit 开发的一款浏览器^①，但从某种角度上来说，它的作用已经远远超越了浏览器，而是成为了一个平台，甚至是一个操作系统。Chrome 继承了 WebKit 内核对 HTML 的高速渲染，同时 Google 自行开发的 V8 引擎使得 JavaScript 在 Chrome 中的执行效率大幅提升，这使得更加高级复杂的 JavaScript 程序在 Chrome 中运行成为可能。

Chrome 浏览器除了页面渲染速度快、JavaScript 执行速度快以外，另一大特点就是支持开发者为其编写各种各样的扩展来扩充其功能，支持用 HTML5 编写桌面程序，这使得 Chrome 变得更加强大。编写这样的程序就是本书所要讲解的内容。

由于 Chrome 扩展和 Chrome 应用有很多相似之处，为了叙述方便，本章会统一说成 Chrome 扩展，但读者应该清楚，这些内容同样适用于 Chrome 应用。

Chrome 扩展是一系列文件的集合，这些文件包括 HTML 文件、CSS 样式文件、JavaScript 脚本文件、图片等静态文件以及 manifest.json。个别扩展还会包含二进制文件，如 DLL 动态库和 so 动态库等，但这需要调用 NPAPI，而 Google 出于安全性考虑，已经决定逐渐淘汰 NPAPI，所以本书不会向大家介绍有关 NPAPI 的内容^②。

安装扩展后，Chrome 就会读取扩展中的 manifest.json 文件。这个文件的文件名固定为 manifest.json，内容是按照一定格式描述的扩展相关信息，如扩展名称、版本、更新地址、请求的权限、扩展的 UI 界面入口，等等。这样 Chrome 就可以知道在浏览器中如何呈现这个扩展，以及这个扩展如何同用户进行交互。

由于 Chrome 扩展是基于 Chrome 平台的，说得直白些，是基于 WebKit 浏览器的——当然有些更加高级的接口不仅仅依赖于 WebKit 浏览器——所以 Chrome 扩展在处理逻辑运算和实现程序功能时，所采用的编程语言必然只能是 JavaScript。

可能你会感到惊讶，毕竟 JavaScript 最开始是为提升网站与用户交互体验而设计的一种轻量

① Chrome 28 之后使用的 Blink 渲染引擎是 WebKit 中 WebCore 组件的一个分支。

② Google 提出了 NaCl（Native Client）取代 NPAPI，感兴趣的读者请自行参见 <https://developer.chrome.com/native-client> 了解相关内容。



级脚本语言，怎么会脱离网站而成为一种程序的逻辑语言呢？随着 Chrome 浏览器 V8 引擎的出现，JavaScript 的执行效率得到了大幅提升，甚至出现了将其作为后端语言的项目——Node.js，所以将 JavaScript 作为一种客户端程序语言就更是绰绰有余了——只要提供更加丰富的功能函数——而 Chrome 平台恰好提供了这样的环境。

总的来说，Chrome 扩展更像是一个运行于本地的网站，只是它可以利用 Chrome 平台提供的丰富的接口，获得更加全面的信息，进行更加复杂的操作。而它的界面则使用 HTML 和 CSS 进行描述，这样的好处是可以在很短的时间内构建出赏心悦目的 UI。界面的渲染完全不需要开发者操心，可以交给 Chrome 去做。同时，由于 JavaScript 是一门解释型语言（现代浏览器使用的 JavaScript 引擎会对 JavaScript 进行编译，如 V8），无需对其配置编译器，因此调试代码时，你只要刷新一下浏览器就可以看到修改后的结果，这使得开发周期大大缩短。

同时，Chrome 浏览器比 Java 虚拟机、Python 解释器（Linux 和 OS X 中默认安装了 Python，而 Windows 中默认没有安装）等其他语言环境更加普及——我甚至可以在我们学校图书馆的计算机中找到 Chrome 浏览器——所以你所开发的 Chrome 扩展可以在更多的计算机中运行。当你遇到一个问题需要利用计算机去处理时，而这台计算机恰好安装了 Chrome 浏览器，那么你就可以欢快地打开记事本开始编写程序了，完成之后加载到 Chrome 浏览器中就可以查看运行结果，这是一件多么酷的事啊！

别急，后面的内容就会让你掌握这项新技能！

1.2 应用与扩展的区别

Chrome 将其平台上的程序分为扩展与应用，并且使用了同样的文件结构，那么两者的区别是什么呢？在早期的 Chrome 版本中，两者的区别非常模糊，而且有些扩展也可以用应用实现，反之亦然。但今天看来，Google 正在努力使两者的界限变得清晰。

总的来说，扩展与浏览器结合得更紧密些，更加强调扩展浏览器功能。而应用无法像扩展一样轻易获取用户在浏览器中浏览的内容并进行更改，实际上应用有更加严格的权限限制，所以应用更像是一个独立的与 Chrome 浏览器关联不大的程序，此时你可以把 Chrome 看成是一个开发环境，而不是一个浏览器。

不过到目前为止，Google 还没有强制规定只能用扩展做什么，只能用应用做什么，所以对于那些扩展和应用都可以实现的功能，到底用何种方式实现，那是开发者自己的选择。不过我建议大家都遵照上述的原则选择实现方式。

除此之外，Chrome 应用还分为 Hosted App（托管应用）和 Packaged App（打包应用），这两者也是有明显区别的。相对而言，Hosted App 更像是一个高级的书签，这种应用只提供一个图标和 Manifest 文件，在 Manifest 中声明了此应用的启动页面 URL，以及包含的其他页面



URL 和这些页面请求的高级权限。比如下面的例子创建了一个启动页面为 `http://mail.google.com/mail/`、包含 `mail.google.com/mail/` 和 `www.google.com/mail/` 且请求 `unlimitedStorage` 和 `notifications` 权限的应用。

```
{
  "name": "Google Mail",
  "description": "Read your gmail",
  "version": "1",
  "app": {
    "urls": [
      "*/mail.google.com/mail/",
      "*/www.google.com/mail/"
    ],
    "launch": {
      "web_url": "http://mail.google.com/mail/"
    }
  },
  "icons": {
    "128": "icon_128.png"
  },
  "permissions": [
    "unlimitedStorage",
    "notifications"
  ]
}
```

Packaged App，顾名思义，就是将所有文件打包在一起的应用，这类应用通常可以在离线时使用，因为它运行所需的全部文件都在本地。

由于 Hosted App 结构和功能都相对简单，所以本书都将重点讲解 Packaged App。

1.3 我的第一个 Chrome 扩展

我发现很多讲解编程的图书，一开始都会详细地讲解相关的预备知识，而大多数读者却更希望马上进行实践。其实，人们总是对基础知识很排斥，这也是为什么教育行业开始推崇自上向下的教材设计方案了——先让读者看到一个最接近表面的东西，之后再慢慢深入地讲解内在的原理和基础。所以我决定在开始讲解之前，先带大家写一个 Demo 程序。这样不仅可以让大家在实践中对基础知识掌握得更加牢靠，同时也调动了大家的积极性。

Chrome 扩展的启动入口可以在浏览器的工具栏和地址栏中，用户单击后激活扩展进行下一步的操作，也可以干脆没有图标，在后台静默地运行。比如微博的扩展，可以设计成将图标显示在工具栏中，用户点击后打开一个显示用户微博时间轴的界面；RSS 订阅器扩展可以设计成将图标显示在地址栏中，当用户点击后，订阅地址栏中当前显示的 URL；自动使用 Google SSL 链接的扩展可以不显示图标，只是在后台默默地监视，当用户访问了非 SSL 的 Google 链接后，



自动跳转到 SSL 链接即可。如图 1-1 所示。

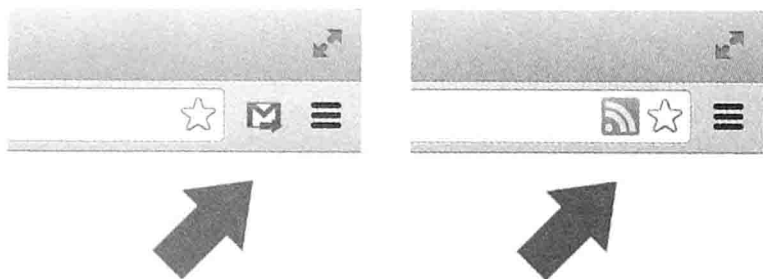


图 1-1 Chrome 扩展图标在浏览器中的位置

我们准备编写一款显示用户计算机当前时间的扩展，这应该比 Hello World 有趣得多。设计思路是在浏览器的工具栏中显示一个时钟的图标，当用户点击这个图标时显示一个实时显示计算机时间的界面。

首先新建一个名为 my_clock 的文件夹，在此文件夹中新建一个名为 manifest.json 的文件，代码如下：

```
{
  "manifest_version": 2,
  "name": "我的时钟",
  "version": "1.0",
  "description": "我的第一个 Chrome 扩展",
  "icons": {
    "16": "images/icon16.png",
    "48": "images/icon48.png",
    "128": "images/icon128.png"
  },
  "browser_action": {
    "default_icon": {
      "19": "images/icon19.png",
      "38": "images/icon38.png"
    },
    "default_title": "我的时钟",
    "default_popup": "popup.html"
  }
}
```

上面的字段中，有些可以一眼看出在定义什么，比如 name 定义了扩展的名称，version 定义了扩展的版本，description 定义了扩展的描述，icons 定义了扩展相关图标文件的位置。version 的值最多可以是由 3 个圆点分为 4 段的版本号，每段只能是数字，每段数字不能大于 6 5535 且不能以 0 开头（可以是 0，但不可以是 0123），版本号段左侧为高位，比如 1.0.2.0 版本比 1.0.0.1 版本更高。每次更新扩展时，新的版本号必须比之前的版本号高。

browser_action 指定扩展的图标放在 Chrome 的工具栏中，browser_action 中的