



DEVTOOLS ▾

多设备 ▾

平台 ▾



应用

扩展

学习基础

**入门教程**

样品

开发扩展

分发扩展

Chrome平台API

帮帮我

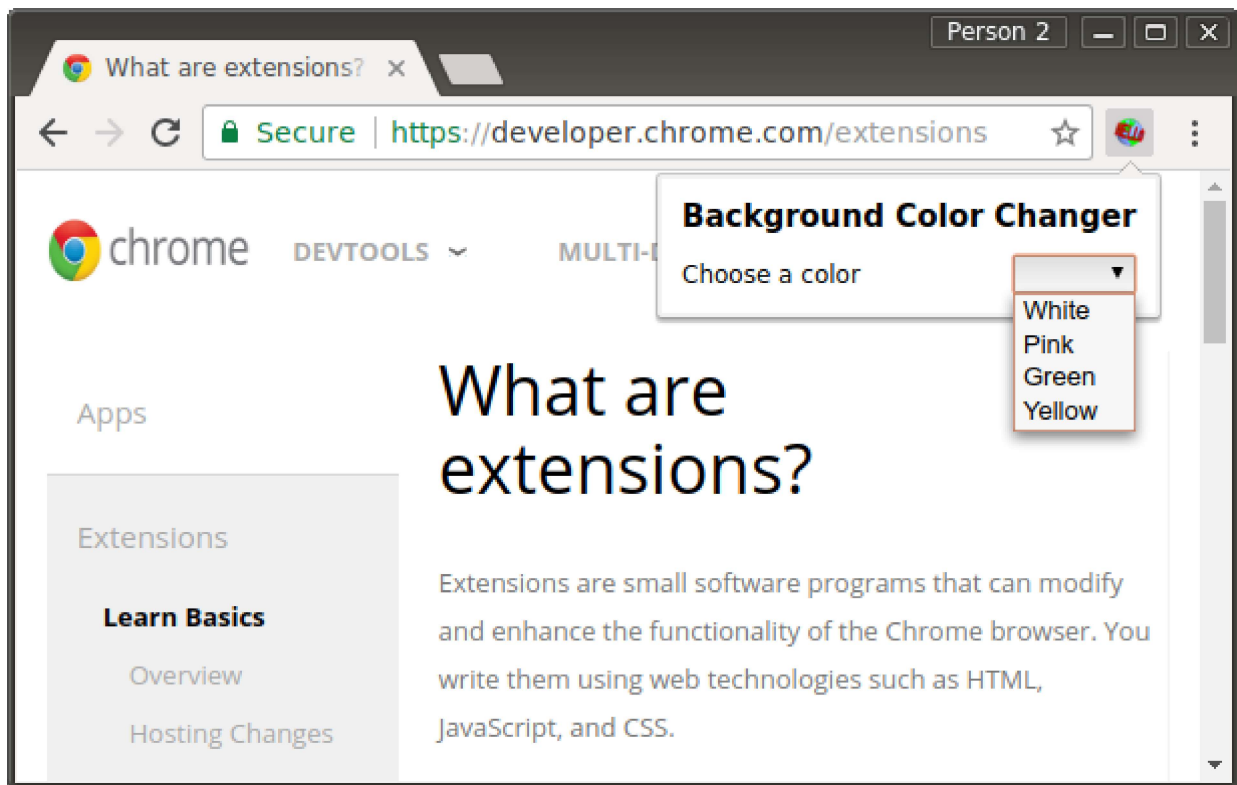
本机客户端

商店

# 入门：构建Chrome扩展程序

通过扩展功能，您可以将功能添加到Chrome，而无需深入研究本机代码。您可以使用您在Web开发中已经熟悉的核心技术为HTML创建新的扩展：HTML，CSS和JavaScript。如果你曾经建立过一个网页，你应该感觉很舒服，我们现在通过构建一个简单的扩展来让用户改变当前网页的背景颜色。

我们将通过实现一个我们称之为[浏览器操作](#)的UI元素来实现这一点，这使得我们可以在Chrome的多功能框旁边放置一个可点击的图标，以便于访问。点击该图标将打开一个弹出窗口，允许用户选择当前页面的背景颜色。如果用户之前已经为页面选择了背景颜色，则一旦弹出被单击，扩展将记住用户的选择并将其用作默认值。以下是它的外观：



如果你想在家里跟随（你应该！），在你的电脑上创建一个闪亮的新目录，并打开你最喜欢的文本编辑器。我们走吧！

## 要声明的东西

我们需要创建的第一件事是名为的清单文件`manifest.json`。这个清单只不过是JSON格式的元数据文件，其中包含扩展名，描述，版本号等属性。在很高的层面上，我们将使用它来向Chrome宣布扩展将要执行的操作，以及为了执行这些操作而需要哪些权限。要了解有关清单的更多信息，请阅读[Manifest File Format文档](#)。

在我们的示例清单中，我们将声明[浏览器操作](#)，[activeTab权限](#)以查看当前选项卡的URL和[存储权限](#)，以记住用户为页面选择的背景颜色。

```
{ "manifest_version" : 2 ,

  "name" : "入门示例" , "description" : "此扩展程序允许用户更改当前页面的背景颜色。" , "version" : "1.0"
,

  "browser_action" : { "default_icon" : "icon.png" , "default_popup" : "popup.html"
} , "permissions" : [ "activeTab" , "storage" ] }
```

继续操作，将这些数据保存到`manifest.json`您创建的目录中的一个文件中，或者 [从我们的样本库 下载一个副本manifest.json](#)。

## 资源

您可能注意到`manifest.json`在定义浏览器操作时指向两个资源文件：`icon.png`和`popup.html`。这两个资源都必须存在于扩展包内，所以我们现在创建它们：

- `icon.png`将会在Omnibox旁边显示，等待用户交互。 [从我们的样本库 下载一个副本icon.png](#)，并将其保存到您正在工作的目录中。如果您倾向于您，也可以创建自己的副本；它只是一个19像素的PNG文件。
- `popup.html`将在为响应用户点击浏览器动作而创建的弹出窗口内呈现。这是一个标准的HTML文件，就像你从web开发中习惯的一样，给你弹出窗口显示的或多或少的空闲状态。 [从我们的示例存储库 下载一份副本popup.html](#)，并将其保存到您正在工作的目录中。




呈现弹出内容的实际逻辑由`popup.js`实现。鼓励您阅读本文件中的注释，以了解更多关于逻辑的信息。 [从我们的示例存储库 下载一份副本popup.js](#)，并将其保存到您正在工作的目录中。

现在你应该在你的工作目录中的四个文件：`icon.png`，`manifest.json`，`popup.html`，`popup.js`。下一步是将这些文件加载到Chrome中。

## 加载扩展

您从Chrome网上应用店下载的扩展程序被打包为`.crx`文件，这对分发很有用，但对于开发来说不是很好。认识到这一点，Chrome为您提供了一个加载工作目录进行测试的快速方法。现在就来做吧

1. `chrome://extensions`在您的浏览器中 访问（或者点击多功能框最右侧的图标打开Chrome菜单，然后在“**更多工具**”菜单下  选择“**扩展**”以访问相同的地方）。
2. 确保选中右上角的**开发者模式**复选框。
3. 点击**加载解压的扩展...**弹出文件选择对话框。
4. 导航到您的扩展文件所在的目录，并选择它。

或者，您可以将您的扩展文件所在的目录拖放到`chrome://extensions`浏览器中以加载它。

如果扩展名是有效的，它将立即加载并激活！如果无效，则会在页面顶部显示一条错误消息。纠正错误，然后重试。

## 用代码捣鼓

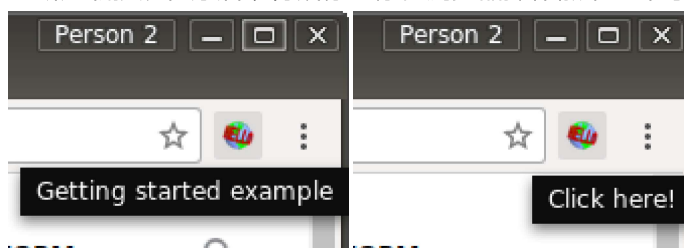
现在您已经完成了第一个扩展，现在让我们摆弄一些东西，以便您了解开发过程的样子。例如，让我们在浏览器操作按钮上设置一个工具提示。

根据browserAction文档，可以通过指定default\_title清单文件中的键来设置工具提示。打开manifest.json，并添加default\_title密钥到 browser\_action。确保JSON是有效的，所以引用键并在必要时添加逗号。

```
{ ... "browser_action" : { "default_icon" : "icon.png" , "default_popup" : "popup.html" , "default_title" : "点击这里!" } , ... }
```

清单文件仅在加载扩展时解析。如果您想查看以前的更改操作，则必须重新加载扩展。访问扩展页面（转到Chrome菜单中的chrome://extensions或More Tools>Extensions），然后点击扩展下的重新加载。所有的扩展也重新加载扩展页面重新加载时，例如击中后F5 or Ctrl-R.

重新加载扩展程序后，将鼠标悬停在浏览器操作徽章上即可看到新的工具提示！



## 接下来是什么？

您现在已经了解了清单文件在把事情集中在一起的中心作用，并掌握了声明浏览器操作的基础知识。这是一个很好的开始，希望能让你有兴趣进一步探索。还有更多的东西在玩。

- “[Chrome扩展程序概述](#)”提供了一些补充，并且大致详细地介绍了扩展的体系结构，以及一些您将要熟悉的特定概念。这是您拓展掌握的最佳下一步。

- 没有人会在第一次尝试中写出完美的代码，这意味着您需要了解可用于调试创作的选项。我们的 [调试教程](#) 是完美的，非常值得仔细阅读。
- Chrome扩展程序可以访问开放网络上可用的强大API：浏览器操作只是冰山一角。我们的[chrome.\\* API文档](#)将依次引导您浏览每个API。
- 最后，[开发者指南](#)还有几十个附加链接，可能会对您感兴趣的文档有所了解。

根据[CC-BY 3.0许可提供的内容](#)

[Google](#) [服务条款](#) [隐私权政策](#) [报告内容错误](#)

 [在上添加我们](#) 