

[home](#) | [javascript](#) [php](#) [python](#) [java](#) [mysql](#) [ios](#) [android](#) [node.js](#) [html5](#) [linu](#)

📄 Chrome 扩展开发——编写一个自己的浏览器插件

[chrome](#) [chrome-extension](#) [javascript](#) [cissy_9](#) 2016年09月20日发布

这次的练习是做一个Chrome的扩展，分享一下入门开发过程。因为在消息传递那块纠结了特别久，所以我会重点总结消息传递那块。

这次做这个插件的功能很简单，就是点击按钮后可以对当前网页的模块进行选择隐藏。
做这个插件一方面是练习实例，还有一方面是，有的时候查资料啊，边上总有很多花花绿绿动来动去的小广告！很烦有木有，还根本不能关闭！就算有关闭按钮，点击了竟然还跳转到广告页面了(°Д°≡°Д°)
所以就想做个小插件，让自己可以选择隐藏这些不想看的模块。

配置文件

每一个扩展都有一个JSON格式的manifest文件，叫manifest.json。

所以第一步我们将创建一个manifest.json文件。如下：

```
{
  "manifest_version": 2, //固定的
  "name": "Cissy's First Extension", //插件名称
  "version": "1.0", //插件使用的版本
  "description": "The first extension that CC made.", //插件的描述
  "browser_action": { //插件加载后生成图标
    "default_icon": "cc.gif", //图标的图片
    "default_title": "Hello CC", //鼠标移到图标显示的文字
    "default_popup": "popup.html" //单击图标执行的文件
  },
  "permissions": [ //允许插件访问的url
    "http://*/",
    "bookmarks",
    "tabs",
    "history"
  ],
  "background": { //background script即插件运行的环境
    "page": "background.html"
    // "scripts": ["js/jquery-1.9.1.min.js", "js/background.js"] //数组.chrome会在扩展启动时
  },
  "content_scripts": [ { //对页面内容进行操作脚本
    "matches": ["http://*/", "https://*/"], //满足什么条件执行该插件
    "js": ["js/jquery-1.9.1.min.js", "js/js.js"],
    "run_at": "document_start", //在document加载时执行该脚本
  } ]
}
```

每个字段的信息我都用注释标明了，接下来就重点说下一些重要字段。

需要注意：

chrome不允许扩展中的HTML页面内直接内嵌js脚本，而要求所有的脚本都作为外部src来引入

browser_action

1. 如果browser action拥有一个popup（即设置了 `default_popup`），popup 可以包含任意你想要的HTML内容，并且会自适应大小。popup 会在用户点击图标后出现。若没有设置 `default_popup`，将执行 `chrome.browserAction.onClicked` 的内容，若没有设置，就什么都不执行了。也就是如果设置了 `default_popup`，就不会执行 `chrome.browserAction.onClicked` 了。
2. 和browser_action对应的还有一个page_action，区别在于：Browser Action对在浏览器中加载的所有网页都生效；Page Action针对特定的网页生效。一个Extension最多可以有一个Browser Action或者Page Action。这里选用Browser Action。

background

1. background是插件的运行环境。若设置了scripts字段，浏览器的扩展系统会自动根据scripts字段指定的所有js文件自动生成背景页。也可以直接page字段，指定背景页。**两者只能设置一个。**
2. 一般情况下，我们会让将扩展的主要逻辑都放在 background 中比较便于管理。其它页面可以通过消息传递的机制与 background 进行通讯。理论上 content script 与 popup 之间也可以传递消息，但不建议这么做。

消息传递

由于插件的js运行环境有区别，所以消息传递机制是一个重要内容。

一次简单的请求

如果仅需要给你自己的扩展的另外一部分发送一个消息（可选的是否得到答复），你可以简单地使用 `chrome.extension.sendRequest()` 或者 `chrome.tabs.sendRequest()` 方法。这个方法可以帮助你传送一次JSON序列化消息从content script到扩展，反之亦然。如果接受消息的一方存在的话，可选的回调参数允许处理传回来的消息。

`sendRequest()` 是Chrome33之前的API，33之后还是使用`sendMessage()`吧。

1. 内容脚本发送消息到扩展程序

```
chrome.extension.sendMessage({hello: "Cissy"}, function(response) {  
    console.log(response.farewell);  
});
```

2. 扩展程序发送消息到内容脚本

```
chrome.tabs.sendMessage(tab.id, {hello: "Cissy"}, function(response) {  
    console.log(response.farewell);  
});
```

3. 接收消息

`chrome.extension.sendMessage()` 向扩展内的其它监听者发送一条消息。此消息发送后会触发扩展内每个页面的 `chrome.extension.onMessage()` 事件。

我用的是长时间的保持连接，原理差不多，就是调用接口的区别，所以就不具体介绍这个了 详细的可以看[开发文档](#)

长时间的保持连接

background 和 popup

同一个Extension的Extension Page (包括background、popup、tab、infobar、notification) 都是运行在同一个进程中的，**所以background 和 popup 之间可以直接相互调用对方的方法，不需要消息传递。**

1. popup调用background中变量或方法

```
var bg = chrome.extension.getBackgroundPage();//获取background页面  
console.log(bg.a);//调用background的变量或方法。
```

2. background调用popup中变量或方法

```
var pop = chrome.extension.getViews({type: 'popup'});//获取popup页面  
console.log(pop[0].b);//调用第一个popup的变量或方法。
```

这里要注意一定要指明 **type** ,如果没有指定，则获取Background Page之外的所有Extension Page的window对象。(。•_•)这个地方真的纠结好久。然后就是background是一个运行在扩展进程中的HTML页面。它在你的扩展的整个生命周期都存在，而popup是在你点击了图标之后才存在，所以，在获取popup变量时，请确认popup已打开。

background 和 content

持续长时间的保持会话需要在content script和扩展建立一个长时间存在的通道。当建立连接，两端都有一个Port对象通过这个连接发送和接收消息。

1. 内容脚本发送消息到扩展程序

```
var bac = chrome.extension.connect({name: "ConToBg"});//建立通道，并给通道命名  
bac.postMessage({hello: "Cissy"});//利用通道发送一条消息。
```

2. 扩展程序发送消息到内容脚本

扩展程序发送消息到内容脚本与前面类似，但需要指定哪个标签需要连接，（获取tab.id的方法我试了很多，但只有下面这个有效，所以如果大家有什么其他有效的方法，求求分享！！）

```
chrome.tabs.onUpdated.addListener(function(tabId, changeInfo, tab) { //获取当前Tab
    var cab = chrome.tabs.connect(tabId, {name: "BgToCon"}); //建立通道, 指定tabId, 并命名
    cab.postMessage({ hello: "Cissy"}); //利用通道发送一条消息。
}
```

3. 接收消息为了处理正在等待的连接, 需要用chrome.extension.onConnect 事件监听器, 对于content script或者扩展页面, 这个方法都是一样的

```
chrome.extension.onConnect.addListener(function(bac) { //监听是否连接, bac为Port对象
    bac.onMessage.addListener(function(msg) { //监听是否收到消息, msg为消息对象
        console.log(msg.hello);
    });
})
```

安装调试

设置 —> 拓展程序 —> 加载已解压的拓展程序 —> 选择文件就行了, 记得要打开开发者模式哦

总结

插件功能的开发我就不写了, 实现起来比较简单, 这篇文章就当是chrome拓展开发的学习笔记了, 不足之处还望指出, 最后还是放一下[插件源码](#)吧, 写的比较乱很多没用到的代码也没删掉, 因为是练习中用到的。嗯嗯好了去吃饭。

参考资料:

[官方文档](#)

[360极速浏览器开发文档](#)

[Chromium扩展 \(Extension\) 机制简要介绍和学习计划](#)

[Chrome插件开发入门 \(二\) —— 消息传递机制](#)

2016年09月20日发布 更多 ▾

0 推荐

收藏

你可能感兴趣的文章

[Webpack下莫名其妙出现的jQuery与报错, 记一次奇妙的Debug旅程](#) 705 浏览


[如何成为一名Chrome应用开发者](#) 27 收藏, 1.2k 浏览

[一个简单的chrome拓展程序开发](#) 2 收藏, 214 浏览



本作品采用 [署名-非商业性使用-禁止演绎 4.0 国际许可协议](#) 进行许可。

2 条评论 默认排序 ▾




南京话zen好听 · 2016年09月22日

源码网盘链接失效了，能给一个github的吗？多谢。

👍 赞

回复




eel_5689479a3768d · 2016年10月12日

下载源码安装后好像插件无法使用啊

👍 赞

回复




文明社会，理性评论

发布评论





App 有新版了？
赶紧下载！！
广告



cissy_9

68 声望

关注作者

发布于专栏

超大布丁

记录一下

2 人关注

关注专栏

相关收藏夹 换一组



开发工具

5 个条目 | 0 人关注



前端文件夹

46 个条目 | 1 人关注



front

11 个条目 | 0 人关注

分享扩散：



Copyright © 2011-2017 SegmentFault. 当前呈现版本 17.02.05
浙ICP备 15005796号-2 浙公网安备 33010602002000号
移动版 桌面版