### **清单文件[#](https://crxdoc-zh.appspot.com/extensions/overview" \l "manifest" \o "Permalink)**

 Manifest.json

{

"name": "我的扩展程序",

"version": "2.1",

"description": "从 Google 获取信息。",

"icons": { "128": "icon\_128.png" },

"background": {

"persistent": false,

"scripts": ["bg.js"]

},

"permissions": [“ "activeTab",”,"http://\*.google.com/", "https://\*.google.com/"],

"browser\_action": {

"default\_title": "",

"default\_icon": "icon\_19.png",

"default\_popup": "popup.html"

}}

"content\_scripts": [{

"matches": [

"http://\*/\*",

"https://\*/\*"

],

"js": [

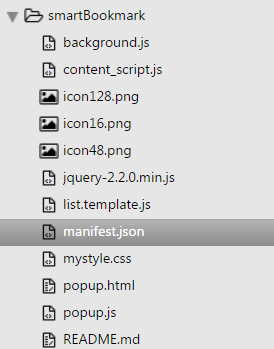
"jquery-2.2.0.min.js",

"content\_script.js"

],

"css": ["mystyle.css"]

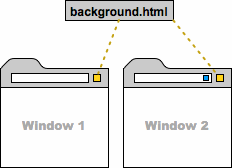
}],

上面的配置文件已经包含了大部分需要开发的文件名： popup.html, background.js, content\_script.js mystyle.css。另外，各种尺寸的icon图片，用于插件在不同地方（插件logo，pop图标，插件商店等等）展示。目录结构如下：  


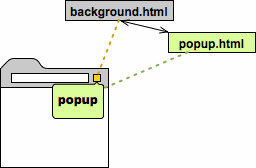
## 架构

### **后台网页**

下图所示的浏览器至少安装了两个扩展程序：一个浏览器按钮（黄色图标）和一个页面按钮（蓝色图标）。浏览器按钮和页面按钮都有后台页面。下图显示了浏览器按钮的后台页面，由background.html 定义，并且包含在这两个窗口中控制浏览器按钮的 JavaScript 代码。

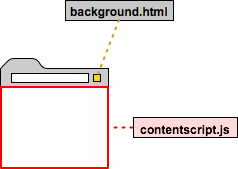


### **用户界面网页**

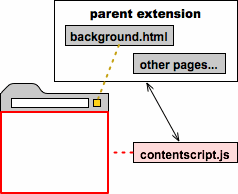


。弹出窗口不用重复后台网页中的代码，因为弹出窗口可以调用后台网页上的函数。

### **内容脚本**



内容脚本并不是完全与所属扩展程序隔离的。内容脚本可以与所属扩展程序交换消息，如下图箭头所示。例如，每当在浏览器页面中发现 RSS 供稿时，内容脚本可以发送消息，反过来后台页面也可以发送消息要求内容脚本更改浏览器页面的外观。



content\_script和background的脚本运行环境不一样，通过message来进行通讯。

## 消息传递

### 一次简单的请求

内容脚本发送消息到扩展程序

chrome.extension.sendMessage({hello: "Cissy"}, function(response) {

console.log(response.farewell);

});

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);

});

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。

**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。

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。

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 。(｡•ˇ‸ˇ•｡)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。

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。

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。

}

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);

})

})

<https://crxdoc-zh.appspot.com/extensions/messaging>

<http://imweb.io/topic/56f7822214ea0f7263803d57>

<https://crxdoc-zh.appspot.com/extensions/getstarted>

<https://crxdoc-zh.appspot.com/extensions/overview#pageComm>