
Google
Developer
Day2009

解析Chrome 浏览器的功能扩展平台（Extension）

Feng Qian（钱锋）

June 2009

Original presentation by Aaron Boodman

Google Developer Day2009

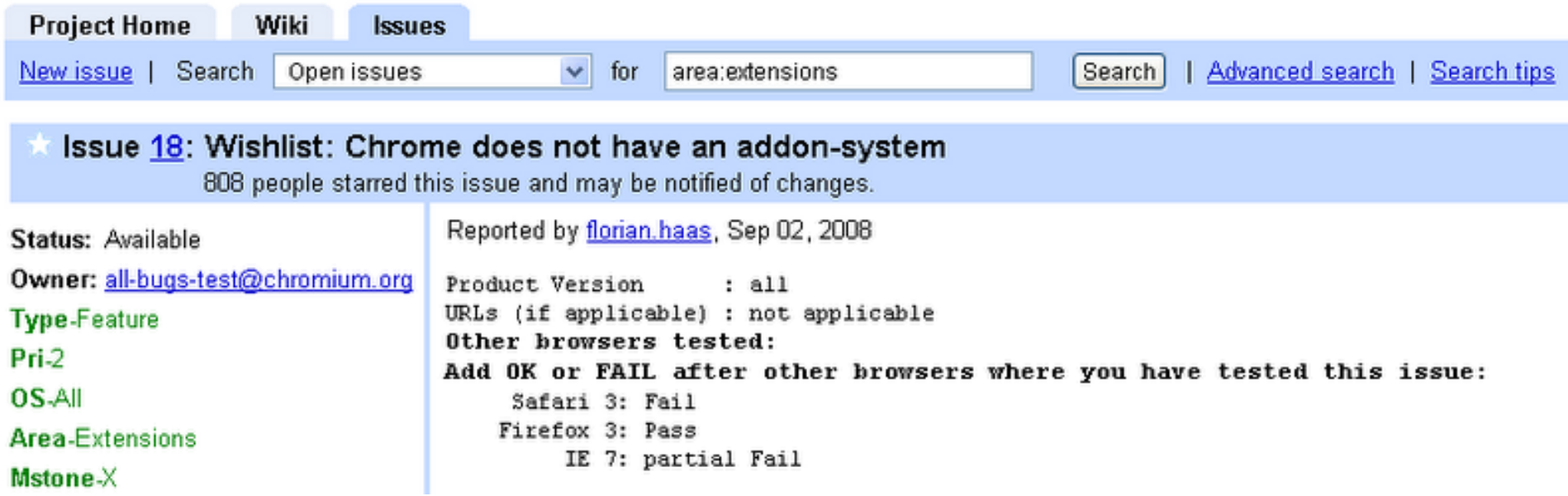
概要

- 介绍
- 进展
- 议程
  - 建立一个功能扩展
  - 介绍Chrome功能扩展很酷的四个方面

为什么要功能扩展？

为什么要功能扩展？

- 用户的热烈要求



为什么要功能扩展？

- 用户的热烈要求

其他好处：

- 保持Chrome最小化
- 为每个人提供一个个性化的浏览器
- 快速实现新功能的原型

进展

- 只有dev channel 或者最近的 trunk 支持功能扩展
- --enable-extensions flag
- Google: Chrome extensions HOWTO 查找指令

1:功能扩展就是网页

HTML, CSS, 和JavaScript

- 一个功能扩展就是由 HTML, CSS, 和 JavaScript 组成的一个压缩文件包
- 功能扩展的每个用户界面部分是一个全功能的网页
- 开放功能扩展就像写网页，可以使用同样的调试工具，同样的JavaScript包，甚至相同的技巧
- 容易的迭代开发周期（iterative development cycle）

演示1:工具条

我们把它们做得好看

```
<div id='button' class='toolstrip-button'>
  <img src='icon.png'>
  <span>Subscribe</span>
</div>
```

- 您可以任意使用CSS的一些小技巧，如果你愿意得话
- 或者使用 WebKit 特有的一些 CSS 扩展

演示2: 使用 CSS

大体上我 们尽量使用Web技术

- 工具条(toolstrips)能根据内容自动改变大小
- 使用 em, 不用 px
- 当窗口大小变化时能从新编排（reflow）

对比

- IE
  - C++, COM
  - 描述性的 WebSlices
- Firefox
  - XUL, JS
  - C++, XPCOM
  - Jetpack (new!)
- Opera
  - HTML Widgets (outside browser)
  - 描述性的 buttons

Chrome

- HTML, CSS, JavaScript (简称 AJAX)
- 也可以用C/C++, 通过NPAPI，但并不鼓励

- “简单的” 功能扩展 API
- 容易

Cross-origin XMLHttpRequest

```
var feedUrl = ...;
var req = new XMLHttpRequest();
rep.onload = handleResponse;
req.open("GET",
        "http://www.google.com/reader/api/0/..." + feedUrl,
        false);
req.send(null);
```

- 和web内容共用cookie jar
- 功能扩展需要需要在清单中声明它要访问的源（origin）

演示3: XMLHttpRequest

浏览器API

- Tabs and windows
- Bookmarks
- Downloads
- etc... (exact list TBD)

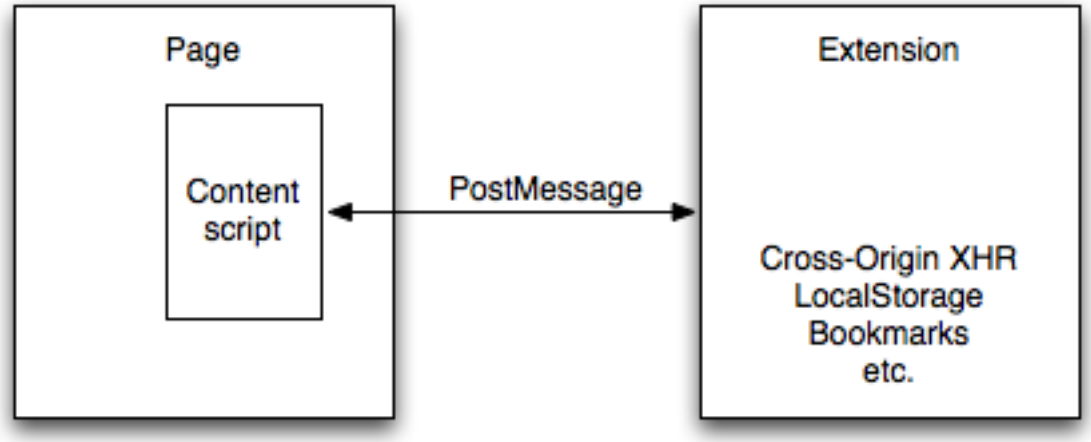
2：内容脚本（content scripts）

内容脚本(content scripts)

```
"content_scripts": [
  {
    "matches": ["http://www.google.com/*"],
    "run_at": "document_start",
    "js": ["foo.js"]
  }
]
```

- 向普通网页中注入JavaScript
- 为了方便性，内容脚本有自己的全局变量
- 目前还没有特殊的权限

内容脚本通信



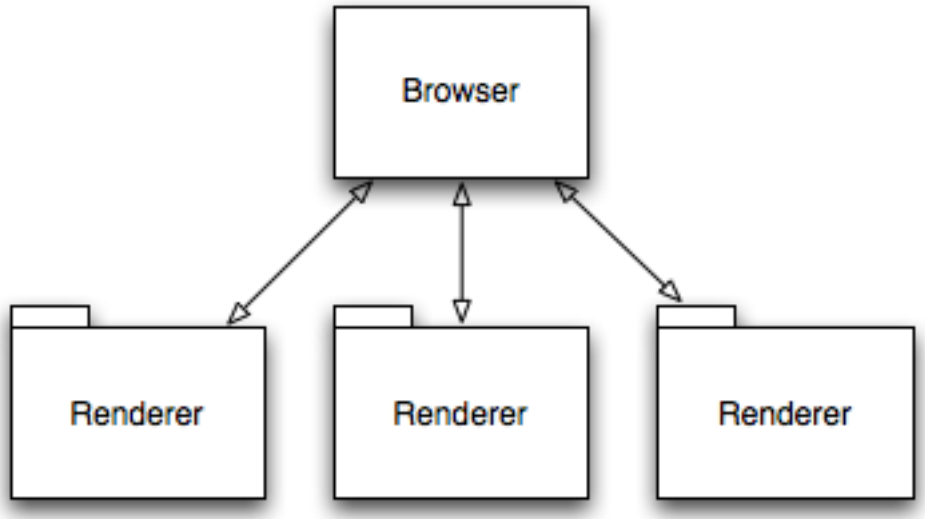

```
// In a content script
var port = chrome.extension.connect();
port.onMessage.addListener(function(data) {
  ...
});
port.postMessage(...);

// In an extension
chrome.self.onConnect.addListener(function(port) {
  port.onMessage.addListener(data) {
    ...
  });
  port.postMessage(...);
});
```

演示4: 内容脚本通信

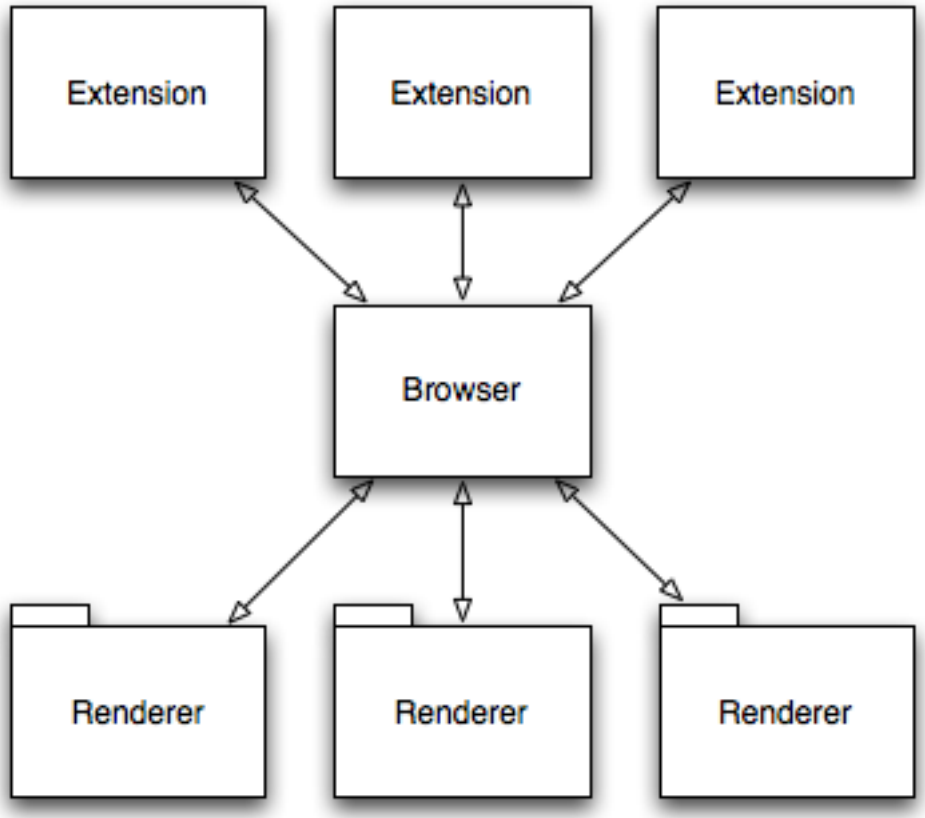
3:功能扩展进程模式

Chrome: 一个多进程浏览器



- 每个tab和plugin有自己的进程
- 网页和plugin无法让整个浏览器奔溃
- Tab程序的漏洞被局限在Tab进程内
- 更好的资源共享

功能扩展有自己的进程



- 一个功能扩展对应一个进程
- 功能扩展无法让整个浏览器奔溃
- 漏洞被局限在功能扩展进程内
- 更好的资源共享

AJAX风格，异步 API

```
chromium.tabs.create(
  { url: "http://www.google.com/" },
  function(tab) {
    alert("Got tab with id: " + tab.id);
  }
);
```

- 多进程要求异步的 API
- 浏览器进程就象个“服务器”
- 使用常见的 AJAX 模式，使的异步编程更容易

一个功能扩展有多个网页构成

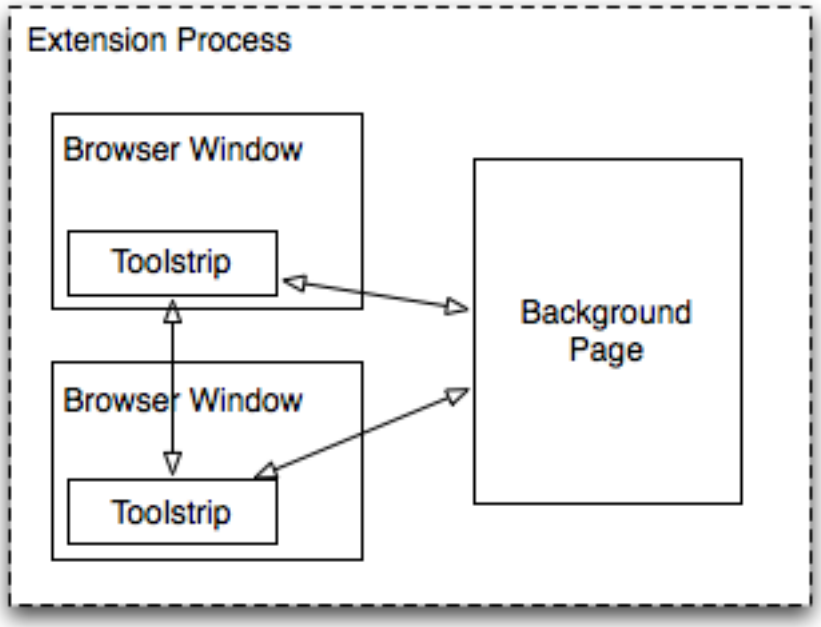
- 每个工具条(toolstrip)，侧栏（sidebar）等等都是一个网页
- 每个浏览器视窗有自己的widget组

功能扩展网页可以相互通信

- 都在同一个进程，同一个线程
- 通信方式和 frame 之间通信相似
- 利用直接的功能调用

```
var total = 0;
chrome.extension.getToolstrips().forEach(function(toolstrip) {
  total += toolstrip.someFunction("foobar");
});
console.log("total is: " + total);
```

一个背景页面把所有的



- 和浏览器窗口无关上下文（context）
- 多数“应用程序代码”在背景页面，工具条和侧栏则更像简单视图（view）

```
button.onclick = function() {
  div.innerHTML = chrome.getBackgroundPage().doSomethingHard();
}
```

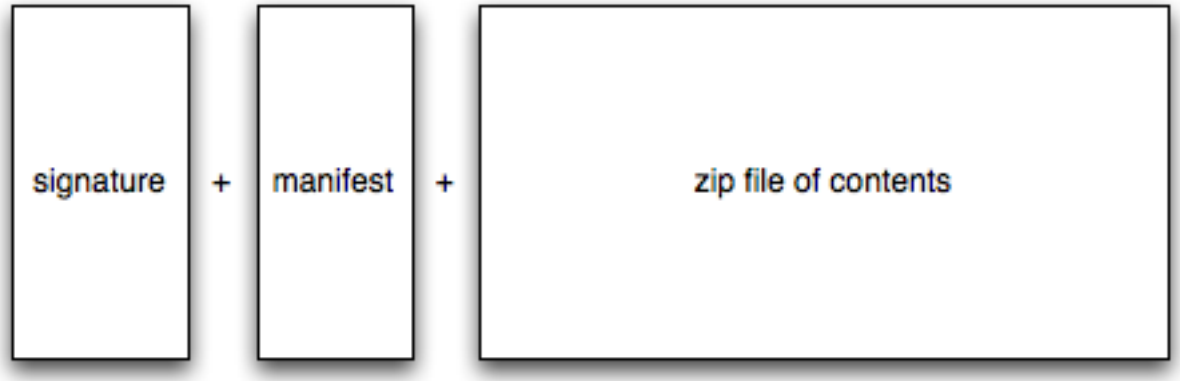
4:打包和分发

manifest.json

```
{
 "id": "subscribe.reader.google.com",
 "name": "Subscribe in Reader",
 "version": "1.0",
 "toolstrips": [
   "mytoolstrip.html"
 ],
 "permissions": [
   { "type": "http", "host": "http://www.google.com" },
   { "type": "tabs" }
 ]
}
```

- 登记功能扩展用到的所有API
- 最少权限的原则

CRX files



- 功能扩展都是经过自我签名来防止中间人攻击（man-in-the-middle attack）
- 文件清单（Manifest） 直接添加到尾部，这样可以快速显示用户界面
- 不要担心钥匙管理，Google将来会提供一项服务使过程变得透明
  - 但是协议和格式很简单，并且是开放的。自己就可以制作CRX文件

部署和安装

- 将CRX文件拷贝到你的服务器
- 即时安装
  - 不需要重启浏览器
- Google 将提供托管服务

更新

- 自动更新
  - 用户不需要任何操作
  - 总是有最新版本
  - 没有重启提示
- 和未来的Chrome版本兼容
- Google 将提供一个易用的更新服务

画廊(Gallery)

- 将来会有一个画廊来展示所有的功能扩展
- 目前还是空的 :)

从这里开始. . .

Google: Chrome Extensions HOWTO

Email: chromium-discuss@googlegroups.com

Q & A


Google
Developer
Day2009