

## 准备工作

需要使用的工具

插件安装与提取

从谷歌商店提取

下载&安装

提取插件安装包

直接下载crx

注意事项！！！

## 修改--解锁高级功能

分析区别

定位判断代码及修改

测试是否成功

## 修改--汉化

分类

例子一：汉化提示语并自定义

例子二：汉化部分设置

例子三：重复部分汉化

其他汉化例子展示

## 修改--布局

给作者填坑

强迫症患者

其他修改（瞎改）

## 修改--杂项

### 修改--动态调试

popup（弹出页面）

植入到页面的js

### 出现错误的处理方法

打包

参考资料

修改样本

# 准备工作

因为很多人需要，而且没有在吾爱破解论坛都没有找到  
那就我来抛砖引玉吧。

## 需要使用的工具

### 1. 脑子。

当然不是说大脑，是思维。

举个例子吧，之前有人问我，为什么把支付失败的字符串改成支付成功后，  
只弹出成功，没有效果。。。这个问题比较深奥，我无法解释。（当时我就说我不会。）

### 2. 正确的上网方式（非必须）

你改插件，总不能改二手的吧。。。

### 3. js , HTML(H5) , CSS 基础(\*)

这些很重要！巧妇难为无米之炊，万丈高楼平地起，离不开。。。

吾爱。

咳咳

#### 4. 习惯用的编辑器。

一个好的编辑器可以让你事半功倍

当然理论上 notepad 也成，就是太费眼。

这里我推荐的两款是 notepad++ 和 sublime

咳咳，至于前者，本来挺喜欢的（只对软件本身），后来事情发生了就果断卸载。。。

如果你确定满足并且想一起研究的话，就继续吧。

正式开始。

## 插件安装与提取

### 从谷歌商店提取

因为我是Chrome的用户（曾经的某6某忠实粉丝hhh劝退）  
理论上 chromium 内核的浏览器插件修改起来应该都差不多吧。

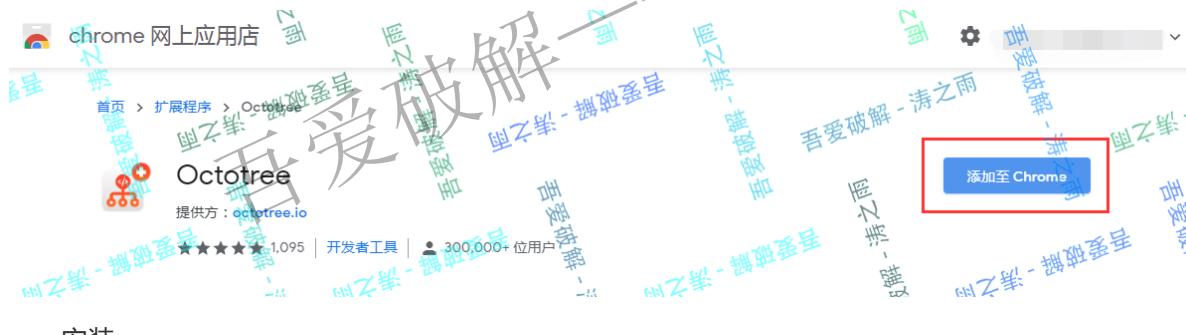
目标：Octotree

官网：[octotree.io](https://octotree.io)

Chrome 插件下载地址（谷歌商店）：[链接太长。。。点我访问](#)

### 下载&安装

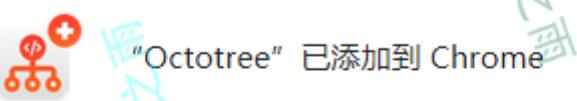
谷歌商店一键安装。



安装



安装完成（感觉这几张都是废图。。。）



点击该图标，即可使用此扩展程序。

您可以通过点击“工具”菜单中的“扩展程序”，管理您的扩展程序。

## 提取插件安装包

chrome://extensions/ 管理页面中，查看刚才安装后的插件的 ID



如图所示，本例的 ID 是 bkhaagjahfmjljalopjnoealnfndnagc（只需要看首尾就足够了。）

好吧，我不知道 Chrome 下载的 crx 文件。。。放在哪里（可能是安装后自动删掉了？）  
有知道的朋友可以回复告知一下。

访问 chrome://version

```

Google Chrome: 83.0.4103.106 (正式版本) (64 位) (cohort: 83_61_Win)
修订版本: ee7134bb3d95141cd18f1e65772a4247f282d950-refs/branch-
heads/4103@{#694}
操作系统: Windows 7 Service Pack 1 (Build 7601.24544)
JavaScript: V8 8.3.110.13
Flash: 32.0.0.371
C:\Windows\system32\Macromed\Flash\pepflashplayer64_32_0_0_37
1.dll
用户代理: Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/83.0.4103.106 Safari/537.36
命令行: "C:\Program Files (x86)\Google\Chrome\Application\chrome.exe"
--no-sandbox --flag-switches-begin --enable-nacl --enable-
webgl-draft-extensions --enable-webgl2-compute-context --
extension-content-verification=bootstrap --disable-smooth-
scrolling --enable-
features=EnableMDRoundedCornersOnDialogs,ParallelDownloading
--flag-switches-end --enable-audio-service-sandbox
可执行文件路径: C:\Program Files (x86)\Google\Chrome\Application\chrome.exe
个人资料路径: C:\Users\      \AppData\Local\Google\Chrome\User Data\Default
其他变体: af01735d-ca7d8d80
dff70c3e-ca7d8d80
317c6f31-377be55a
7a26526b-ca7d8d80
87463348-cbc25d7a
4ca682fe-71d5c685
273268f8-f820de84

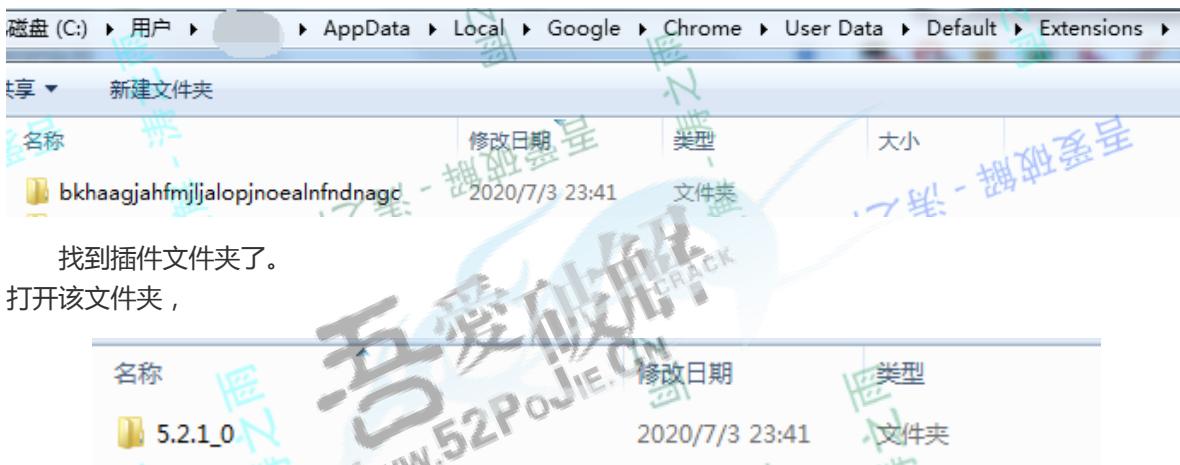
```

找到 个人资料路径，计算机里打开路径。

插件所在文件夹在 Chrome 资料文件夹下的 Extensions 文件夹

比如本例：

看首几位为：bkha



找到插件文件夹了。  
打开该文件夹，

里面这个版本号所在的文件夹才是真正的插件文件夹（插件是需要打包才能成为 crx 文件的，本质上就是有经过签名压缩文件，后面会说怎么签名打包）

复制这个版本号文件夹出来。

## 直接下载crx

有些浏览器或某些插件可以直接下载到 crx 或 未封装的插件

下载后直接解压就好。

## 注意事项！！！

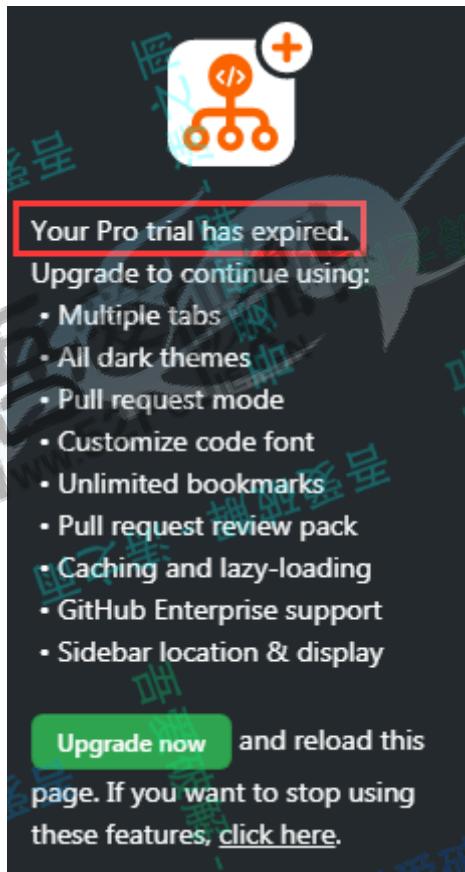
注意！和爆破exe一样！最重要的就是勤备份！  
注意！和爆破exe一样！最重要的就是勤备份！  
注意！和爆破exe一样！最重要的就是勤备份！  
不然一失足成千古恨。。。

## 修改-解锁高级功能

获取到目标文件的解压文件就可以开始着手了  
推荐的参考

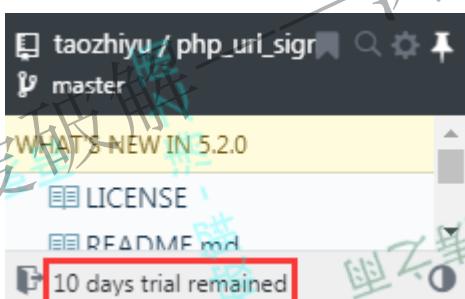
## 分析区别

破解功能自然是因为功能上有差异。找到这个差异。  
比如



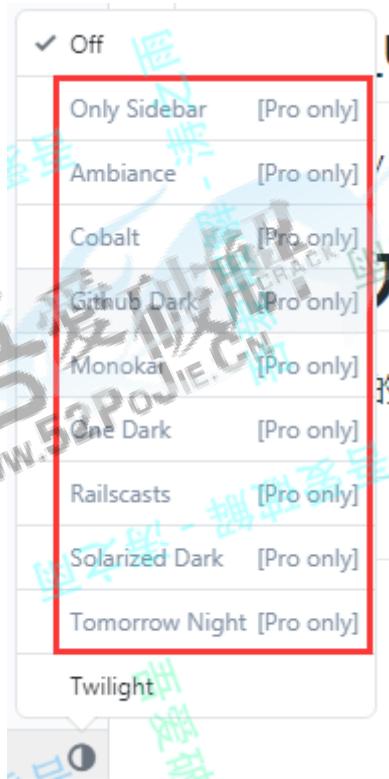
| 试用会员过期提示升级的提示 (源码里称之为广告)

或是



| 剩余会员试用时间的显示

亦或是



仅会员可以使用的提示。

当然还有很多情况，比如只有会员才会显示的按钮（官方示意图可能会有关键词）

## 定位判断代码及修改

推荐 notepad++ 和 sublime 之一的原因就是他们支持文件夹中搜索关键词（好吧，everything好像也可以，不过那不是编辑器。。。）

例如 试用会员还剩多长时间过期 的提示

关键词：remained，文件夹内搜索，发现只有 content.js 里有。

```
Searching 32 files for "remained" (regex)
Q:\chrome_plugin\Octotree\5.2.1_0\content.js:
中间省略。。
1 match in 1 file
```

打开继续搜索

```
uebar.find(".octotree-ads-big").remove();_backToSidebarView();($ ".octotree-auth-view").remove(),this._authView=$([]);  
buildRegisteredView(){this._optsView.$view.append(p).buildProv({email:e,trialRemainingDays:a});const  
i="octotree-trial-msg-view",o=".octotree-user-profile-view";if(  
this._state==t.SUBSCRIPTION_OK&&this._treeView.$view.append(f),a.$(o).remove();let e=this._sidebar.find(i);if(  
0!==e.length)return e;this._authFooter.prepend(d).find(".octotree-trial-footer_logout").click(  
this._onLogoutClick.bind(this)),(e=this._sidebar.find(i)).text(`${a} ${a1?"days":"day"} trial  
remained`)else{$(i).remove();let t=this._sidebar.find(o);if(0!=t.length)return t;this._authFooter.prepend(  
i),(t=this._sidebar.find(o)).find("i").click(this._onLogoutClick.bind(  
this)),t.find(".octotree-user-profile-view_email a").text(e)}async _verifySubscription(e){const i=e||await  
extStore.get(a),o=this._decodeTokenPayload(i);let s=t.UNAUTHENTICATED;if(  
o){const _expiredAt=_subscriptionStatus.a._trialDays_createdAt+n;a-if(a){switch(  
a){case "P":return s.unauthenticated;case "A":return s.accessed;case "C":return s.accessed;case "S":return s.accessed;case "D":return s.accessed;case "E":return s.accessed;case "F":return s.accessed;case "G":return s.accessed;case "H":return s.accessed;case "I":return s.accessed;case "L":return s.accessed;case "M":return s.accessed;case "N":return s.accessed;case "O":return s.accessed;case "R":return s.accessed;case "T":return s.accessed;case "U":return s.accessed;case "V":return s.accessed;case "W":return s.accessed;case "X":return s.accessed;case "Y":return s.accessed;case "Z":return s.accessed;}}}}})
```

发现只有一处（巧了，看一下前后文，应该是的 day(s) 也有了）

## 向前找判断

这里因为打包的时候压缩了，建议在新窗口美化一下（当然也可以直接修改。或是直接保存美化后的代码。个人喜欢原汁原味）

当然有些网站美化后的代码会出现问题。。比如

```
Multiple Github accounts ","
GitHub Enterprise support ","
And many more...
"].map(e=> <li><span>${e}</span></li>`).join(" ")
"),r='`  
          <n  <div class="octo-tree - outage - view "><n  Server unavailable. Please <a class="octo-tree - outage - view _reload " href="try again"></a>.<n  </div>`  
          ,l='`  
          <n  <div class="octo-tree - user - profile - view "><n  <div class="octo-tree - user - profile - view _logout "><n  <span class="tooltipped tooltiped - ne " aria-label="Logout "><n  <i /><n  </span><n  </div>`  
          <n  <div class="octo-tree - user - profile - view _email "><n  <a href="https://www.octo-tree.io/subscription" target=_blank"><n  </a><n  </div>`  
          <n  <div class="octo-tree-auth-view-login"><n  <div class="octo-tree-auth-view-login__logo"><n  <span></span><n  </div>`  
          <n  <form autocomplete="off"><n  <div style="display: none;"><n  <input type="text" id="PreventChromeAutocomplete" name="PreventChromeAutocomplete" autocomplete="address-level4" /><n  </div>`  
          <n  <div><n  <input type="email" name="email" class="form-control" placeholder="Enter email" autocomplete="nope"/><n  </div>`  
          <n  <div><n  <input type="password" name="password" class="form-control" placeholder="Enter password" autocomplete="new-password"/><n  </div>`  
          <n  <div class="octo-tree-auth-view-login__submit"><n  <button type="submit" class="btn btn-primary">Login</button>`  
          <n  <a href="https://www.octo-tree.io/register"
```

这样。

因为瞎换行，把人家的字符串给拆了。。。。

```
6709 _buildProView({ email: e, trialRemainingDays: a }) {
6710   const l = ".octotree-trial-msg-view",
6711         o = ".octotree-user-profile-view";
6712   if (this._state === t.SUBSCRIPTION_OK && this._treeView.$view.append(f), a) {
6713     f(o).remove();
6714     let e = this._$sidebar.find(i);
6715     if (0 !== e.length) return e;
6716     this._$authFooter.prepend(d).find(".octotree-trial-footer__logout").click(this._onLogoutClick.bind(
6717       this)), (e = this._$sidebar.find(i)).text(`{$a} ${a} days`)"day" trial remained`)
6718   } else {
6719     $(i).remove();
6720     let t = this._$sidebar.find(o);
6721     if (0 !== t.length) return t;
6722     this._$authFooter.prepend(l), (t = this._$sidebar.find(o)).find("i").click(this._onLogoutClick.bind(
6723       this)), t.find(".octotree-user-profile-view__email a").text(e)
```

格式化后看起来就很（相对）清爽了。

一个方法，里面是判断。

根据判断找到

```
this._state === t.SUBSCRIPTION_OK
```

此处其实可以使用全局替换，把 `this._state === t.SUBSCRIPTION_OK` 改成 `true`。

此处选择方法分析，尽可能少改动。

搜索调用（这个没有混淆，很方便找。如果混淆了一般都会动态调试，下面有动态调试的方法，可以参考一下）

```
if (this._optsView = 1, this._authView = $([ ]), this._sidebar = d, this._treeView = c, this._document = h, this._state === t.SERVER_OUTAGE) return this._buildOutageView(), void(this._state = null);
const p = await extStore.get(a);
if (!p) return await this._cleanUpPlugins(), void this._buildLiteView(e, s);
this._state = await this._verifySubscription(p), await this._cleanUpPlugins();
const _ = this._decodeTokenPayload(p);
switch (this._state) {
    case t.UNAUTHENTICATED:
        this._buildLiteView(e, s);
        break;
    case t.TRIAL_EXPIRED:
    case t.SUBSCRIPTION_EXPIRED:
    case t.SUBSCRIPTION_OK:
    {
        const a = this.getTrialRemainingDays(_);
        0 === a && this._state !== t.SUBSCRIPTION_OK && (this._buildRegisteredView(),
            await extStore.get(o) || this._openTrialAds()),
        this._buildProView({ email: e.email, trialRemainingDays: a });
        const i = this._state === t.SUBSCRIPTION_OK ? [...s, ...n, ...r] : [...s, ...n];
        await this._activatePlugins({ plugins: i, inputs: e, payload: { state: this._state } });
        break
    }
}
i.final || (async () => {
    const e = await this._validateAndSyncSubscriptionState();
    this._state !== e && (this._state = e, this.reload())
})()
```

这里一样，因为很轻松就能知道 `SUBSCRIPTION_OK`、`SUBSCRIPTION_EXPIRED`、

`TRIAL_EXPIRED` 是什么意思，直接改就行了。如果混淆了，需要动态调试。

搜索后得知 `_getTrialRemainingDays` 只有一处调用，因此直接赋值就好，不会有问题是。

```
1 | const a = this._getTrialRemainingDays();
```

改成

```
1 | const a = 0; // 此处改成0（试用剩余0天，试用期间有购买Pro的广告）
```

接着是改 `this._state`，因为多次调用该代码，因此溯源。找 `this._state` 赋值的位置。  
就在上面一行（格式化后的）

```
const p = await extStore.get(a);
if (!p) return await this._cleanUpPlugins();
this._state = await this._verifySubscription(p);
await this._cleanUpPlugins();
const t = this._decodeTokenPayload(p);
switch (this._state) {
    case t.UNAUTHENTICATED:
```

```
1 | async _verifySubscription(e) {
2 |     const i = e || await extStore.get(a),
3 |         o = this._decodeTokenPayload(i);
4 |     let s = t.UNAUTHENTICATED;
5 |     if (o) {
6 |         const { expiredAt: e, subscriptionStatus: a, trialDays: i, createdAt: n } = o;
7 |         if (Date.now() < e)
8 |             if (a) switch (a) {
9 |                 case "EXPIRED":
10 |                     s = t.SUBSCRIPTION_EXPIRED;
11 |                     break;
12 |                 case "ACTIVE":
13 |                 case "CANCELLED":
14 |                     s = t.SUBSCRIPTION_OK
15 |             } else s = Math.ceil((Date.now() - n) / 864e5) <= i ?
16 |                 t.SUBSCRIPTION_OK : t.TRIAL_EXPIRED
17 |             }
18 |     return s === t.UNAUTHENTICATED && await extStore.remove(a), s
}
```

老规矩，可以把判断改掉。

我这里把除了 `SUBSCRIPTION_OK` 的赋值全都改了。

```
1 | async _verifySubscription(e) {
2 |     const i = e || await extStore.get(a),
3 |         o = this._decodeTokenPayload(i);
4 |     let s = t.UNAUTHENTICATED;
5 |     if (o) {
6 |         const { expiredAt: e, subscriptionStatus: a, trialDays: i, createdAt: n } = o;
7 |         if (Date.now() < e)
8 |             if (a) switch (a) {
9 |                 case "EXPIRED": // 把下面两行过期的跳转删掉
10 |                 case "ACTIVE":
11 |                 case "CANCELLED":
12 |                     s = t.SUBSCRIPTION_OK
```

```

13         } else s = Math.ceil((Date.now() - n) / 864e5) <= i ?
14             t.SUBSCRIPTION_OK : t.SUBSCRIPTION_OK //这里把判断前后改成一样的就成
15     }
16 }

```

当然全都删掉然后改成

```
1 | return t.SUBSCRIPTION_OK
```

也是可以的。

我个人习惯基于原代码修改。

## 测试是否成功

把上面的代码在原文件里对应修改。  
保存后点击



重新加载插件就修改完成了。

这里是成功破解了高级功能。

如果不成功返回继续

如果报错见[出现错误的处理方法](#)

## 修改--汉化

### 分类

上一次修改的时候对代码进行了汉化，这次也不例外。  
汉化的情况有多重。

1. 语言是写死的（类似这次，是只有单一语言的）
2. 有多重语言包可以选择，就是没有中国的（差评）
3. 有中文的选项，但是只有部分是汉语，而且还有好多是错的。

一般来说，

- 1 是因为没做多语言配置，在js里直接修改就好
- 2 相对来说好改一点，找到语言包，修改就行（一般是json文件或是js内的json，且多为谷歌语言包，在插件目录下的\_locales文件夹下，有对应语言的语言包）
- 3 一般和2情况类似，补充翻译即可。（推荐BCompare 4，对比很方便，其他某插件的示意图如下图）

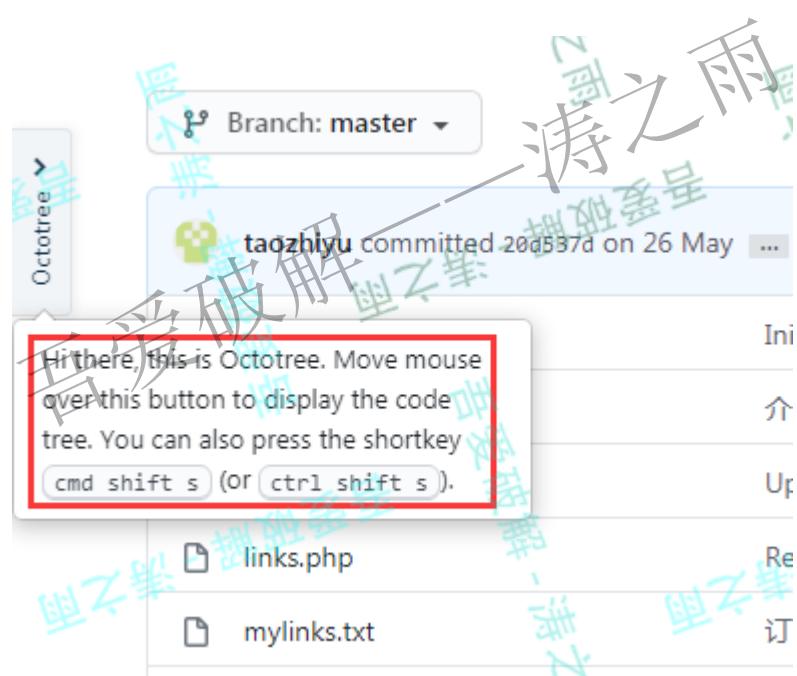
2020/4/4 0:38:00 42,728 字节 其它一切 UTF-8 PC

```
648     "message": "关闭除当前活动的目录之外的所有目录。"
649 },
650
651     "settings_directories_icon_size": {
652         "message": "目录图标大小"
653     },
654     "settings_directory_arrows": {
655         "message": "目录箭头"
656     },
657
658     "settings_expert_save_error": {
659         "message": "无法保存设置。请更正您的输入。"
660     },
661     "settings_expert_settings_desc": {
662
663         "message": "Closes all directories except the currently active one."
664     },
665     "settings_dir_open_duration": [
666         "message": "Directory open duration"
667     ],
668     "settings_dir_open_duration_desc": {
669         "message": "Duration of the open and close animation of directories"
670     },
671     "settings_directories_icon_size": {
672         "message": "Directories icon size"
673     },
674     "settings_directory_arrows": {
675         "message": "Directory arrows"
676     },
677     "settings_dnd_creation_dialog": [
678         "message": "Show creation dialog"
679     ],
680     "settings_dnd_creation_dialog_desc": {
681         "message": "Show the creation dialog before adding a new bookmark when drag&drop"
682     },
683     "settings_dnd_open_dir_delay": [
684         "message": "Drag&Drop Open directory delay"
685     ],
686     "settings_dnd_open_dir_delay_desc": [
687         "message": "Delay before opening a hovered directory when dragging an element"
688     ],
689     "settings_expert_save_error": [
690         "message": "Could not save settings. Please correct your input."
691     },
692     "settings_expert_settings_desc": {
```

本插件是最不方便修改的写死在js里的类型。一般需要所见即需译，就是看到需要翻译的，就去翻译，而且翻译的过程最好从特征词句到一般词语。  
有强烈特征的一句话或是一个不常见的单词先翻译。比如 `settings`，一般只要有设置的插件里会反复出现，这样翻译会方便。

## 例子一 · 汉化提示语并自定义

举个例子



第一次安装完后打开会有如下提示，一句话，就很方便搜索到

```
const e = '<div>\n  <nav class="octotree-sidebar">\n    <div class="octotree-toggle">\n      <i class="octotree-toggle-icon" role="button"></i> Octotree</span>\n    <div class="popups">\n      <div class="arrow"></div>\n      <div class="content">\n        Hi there, this is Octotree.\n        Move mouse over this button to display the code tree. You can also press the\n        shortkey <kbd>cmd shift s</kbd> (or <kbd>ctrl shift s</kbd>).\n      </div>\n    </div>\n  </div>\n</div>\n<div class="octotree-main-icons">\n  <a class="octotree-pin" data-store="PINNED">\n    <span class="tooltipped tooltipped-s" aria-label="Pin this sidebar">\n      <i class="octotree-icon-pin"></i>\n    </span>\n  </a>\n  <a class="octotree-settings">\n    <span class="tooltipped tooltipped-s" aria-label="Settings">\n      <i class="octotree-icon-settings"></i>\n    </span>\n  </a>\n</div>\n<div class="octotree-views">\n  <div class="octotree-view octotree-tree-view current">\n    <div>
```

那就直接修改好了。注意不要修改到关键词，或是其他代码，或是单双引号，亦或是%1、\$1、{1} 这类的替换字符。

翻译替换进去就行，这里我选择[谷歌翻译](#)，可能是日常习惯用谷歌搜索吧，而且比较喜欢 Material Design 风格吧。



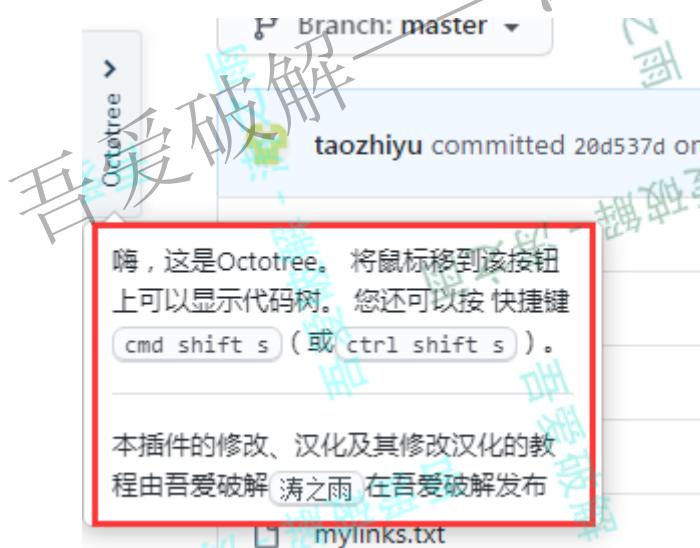
虽然都是简单的句子，我还是选择机译+校准。因为打字没有复制粘贴快（**所以码教程是最痛苦的，不接受反驳**）

当然，你也可以随手扔一个“版权”之类的。。。虽然破解版只是技术的产物，是没有版权可验的。

修改完：

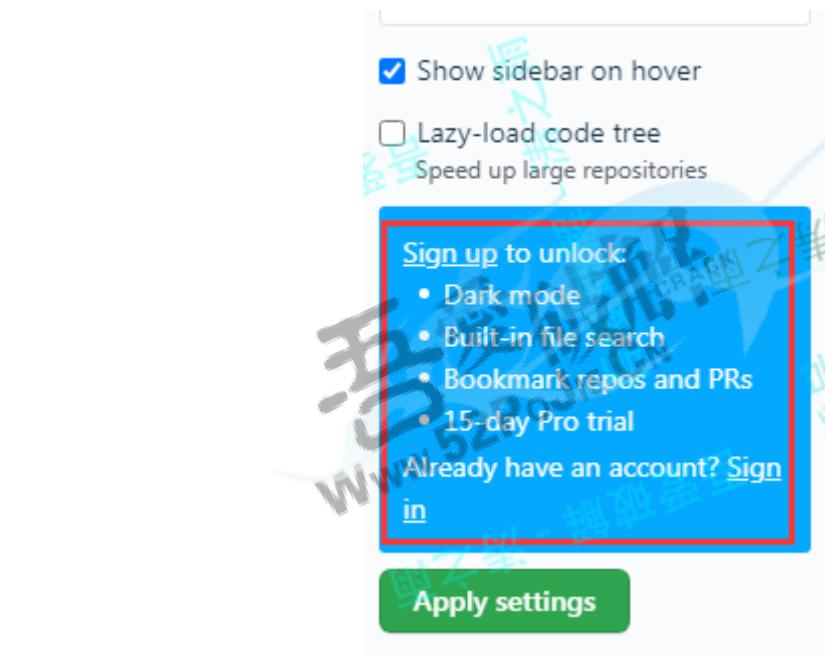
```
class="octotree-toggle-icon" role="button"></i> <span>Octotree</span>\n\n <div class="popup">\n <div class="arrow"></div>\n <div class="content">\n 嗨，这是Octotree。将鼠标移到该按钮上可以显示代码树。您还可以按\n 快捷键<kbd> cmd shift s </kbd> (或<kbd> ctrl shift s </kbd>)。<br/>本插件的修改、汉化及其修改汉化的教程由吾爱破解<kbd>涛之雨</kbd>在吾爱破解发布</div>\n </div>\n </div>\n\n <div class="octotree-main-icons">\n <a class="octotree-pin" data-store="PINMED">\n <span>
```

效果图：



## 例子二：汉化部分设置

再举个例子



因为我刚才卸载了，提示我要登录。

改之，老规矩，搜索上面的关键词 unlock：

```
such as multiple tabs, code font settings, code review enhancement and performance improvement\n </div>\n <div class="octotree-auth-view-login_error"></div>\n </form>\n </div>\n </div>\n',\n d = '\n <div class="octotree-trial-footer">\n <span class="octotree-trial-footer_logout tooltiped tooltiped-ne" aria-label="Logout">\n <i />\n </span>\n <div class="octotree-trial-msg-view">\n </div>\n',\n h = '\n <div class="octotree-ads">\n <a target="_blank" href="https://www.octotree.io/register">Sign up</a>\n to unlock:<ul>\n ${s}</ul>\n Already have an account? <a class="login-btn">Sign in</a>\n </div>\n',\n p = '\n <div class="octotree-ads">\n <a target="_blank" href="https://www.octotree.io/checkout">Upgrade to\n Pro</a>\n to unlock:<ul>\n ${n}</ul>\n <a target="_blank" href="https://www.octotree.io">Learn\n more</a>\n </div>\n',\n _ = '\n <a class="login-action">Login to unlock more features</a>\n',\n u = '\n <div class="octotree-ads octotree-ads-big">\n <i class="octotree-icon-x"></i>\n Enjoying Octotree?\n <a target="_blank" href="https://www.octotree.io/register">Sign up</a>\n to unlock these features:<ul>\n ${s}</ul>\n If you already have an account, please <a class="octotree-ads-big login-btn">sign in</a>\n
```

搜索后发现并没有解锁的列表，在 \${s} 和 \${n} 里。

向上翻一下，找到了该列表。

```
i = 'octotree.ats',\n o = 'octotree_trial-confirm',\n s = ["Dark mode", "Built-in file search", "Bookmark repos and PRs", "15-day Pro trial"].map(e => `<li><span>${e}</span></li>`).join(""),\n n = ["Multiple tabs", "More dark themes", "Pull request mode", "Faster performance", "Customize code font", "Unlimited bookmarks", "Multiple Github accounts", "GitHub Enterprise support", "And many more..."].map(e => `<li><span>${e}</span></li>`).join(""),\n r = '\n <div class="octotree-outage-view">\n Server unavailable. Please <a
```

既然是破解版，那就可以随心所欲的翻译了。。。比如我是这么做的



对应的代码部分：

### 例子三：重复部分汉化

汉化 settings 设置的标题部分。有很多搜索项



可以

1. 勾选 Aa 区分大小写。
  2. 使用正则表达式，排除前后没有的字符比如使用

1 | \wsettings\w

使用正则表达式，排除前后没有的字符比如使用

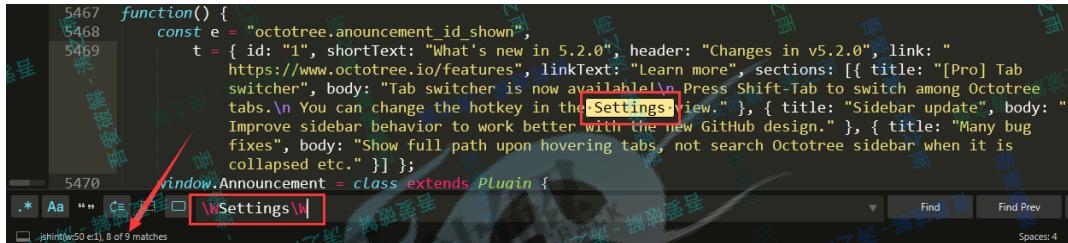
WSEttings (w)

吾爱破解

第 13 页 共 23 页

<https://www.52pojie.cn>

### 匹配前后都是非字母和数字的 settings



```

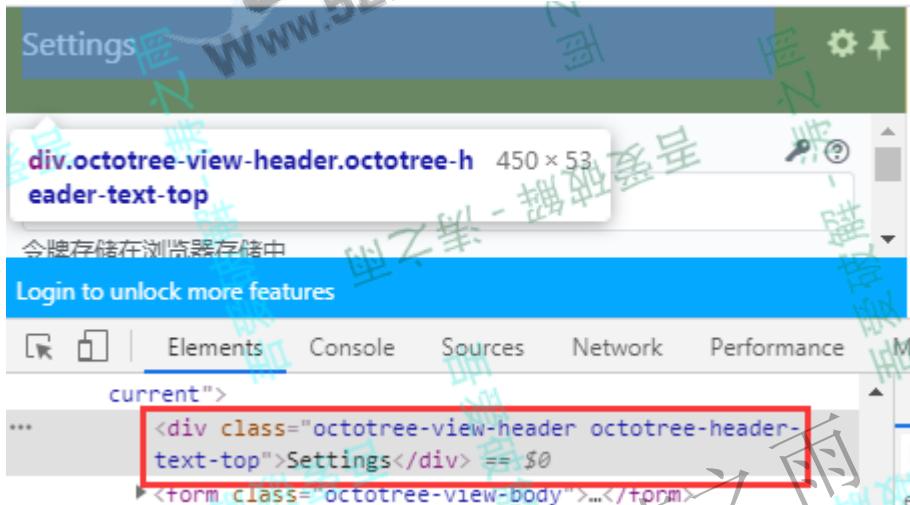
5467 function() {
5468   const e = "octotree.announcement_id_shown",
5469     t = { id: "1", shortText: "What's new in 5.2.0", header: "Changes in v5.2.0", link: "
https://www.octotree.io/features", linkText: "Learn more", sections: [{ title: "[Pro] Tab
switcher", body: "Tab switcher is now available! Press Shift-Tab to switch among Octotree
tabs.\n You can change the hotkey in the Settings view.", }, { title: "Sidebar update", body: "
Improve sidebar behavior to work better with the new GitHub design.", }, { title: "Many bug
fixes", body: "Show full path upon hovering tabs, not search Octotree sidebar when it is
collapsed etc." }] };
5470 window.Anouncement = class extends Plugin {
...

```

(这9处除了一个夜间模式匹配字符的正则表达式，其余的都是要汉化的)

### 3. 审查元素，找到前后文（包括类名、ID）

比如



很方便就可以精准定位到汉化的位置。

```

<octotree-tree-view current>\n <div class="octotree-view-header"></div>\n <div
class="octotree-view-body"></div>\n <a class="octotree-spin"> <span
class="octotree-spin--loader"></span> </a>\n </div>\n <div class="octotree-view
octotree-error-view">\n <div class="octotree-view-header octotree-header-text-top"></div>\n <
form class="octotree-view-body"><div class="message"></div></form>\n </div>\n <div
class="octotree-view octotree-settings-view">\n <div class="octotree-view-header
octotree-header-text-top">Settings</div>\n <form class="octotree-view-body">\n <div>\n <label>
GitHub access token</label>\n <div class="octotree-token-actions">\n <a
class="octotree-create-token" target="_blank" tabindex="-1">\n <span class="tooltipped
tooltipped-w" aria-label="Generate new token">\n <i class="octotree-icon-key"></i>\n </span>\n
</a>\n <a class="octotree-help" href="https://www.octotree.io/features#access-token">\n

```

## 其他汉化例子展示

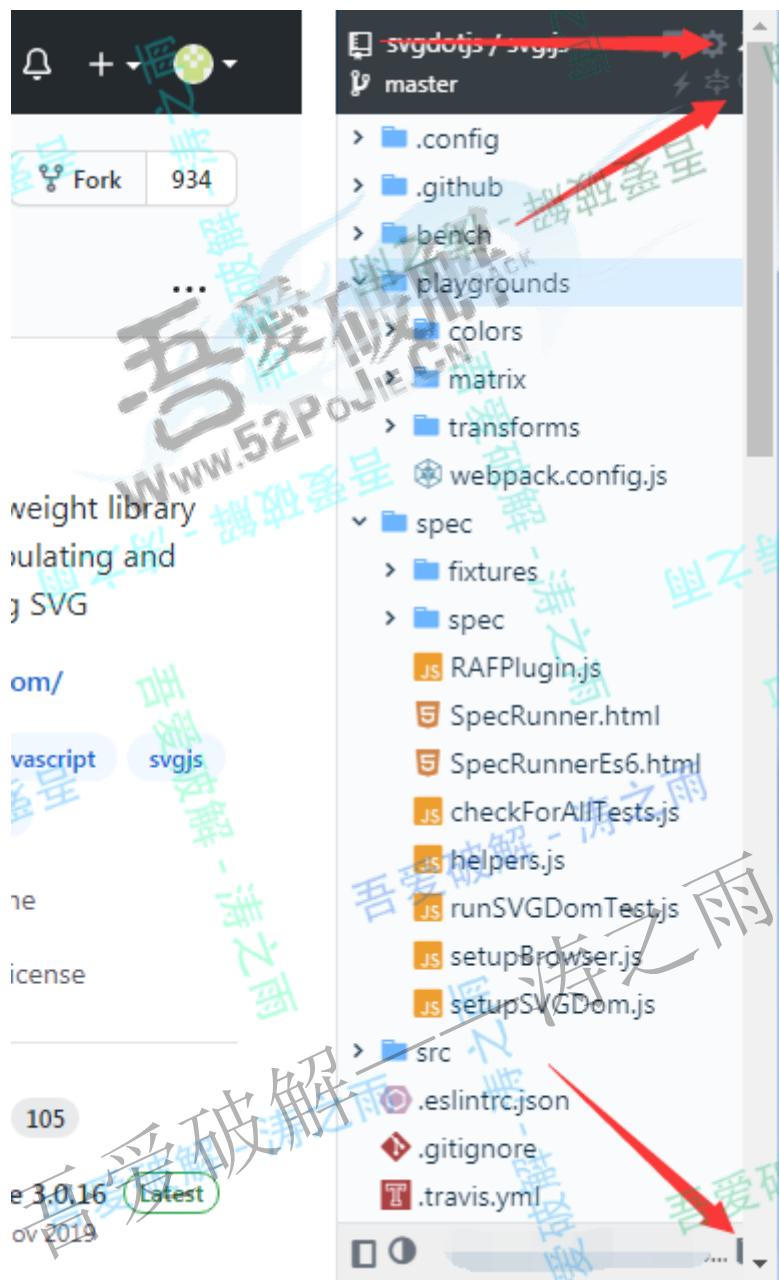
请下载插件（同时提供了修改后的和未修改的版本）

## 修改--布局

布局一般修改是 CSS，JS 甚至 HTML，因此要都有所了解。

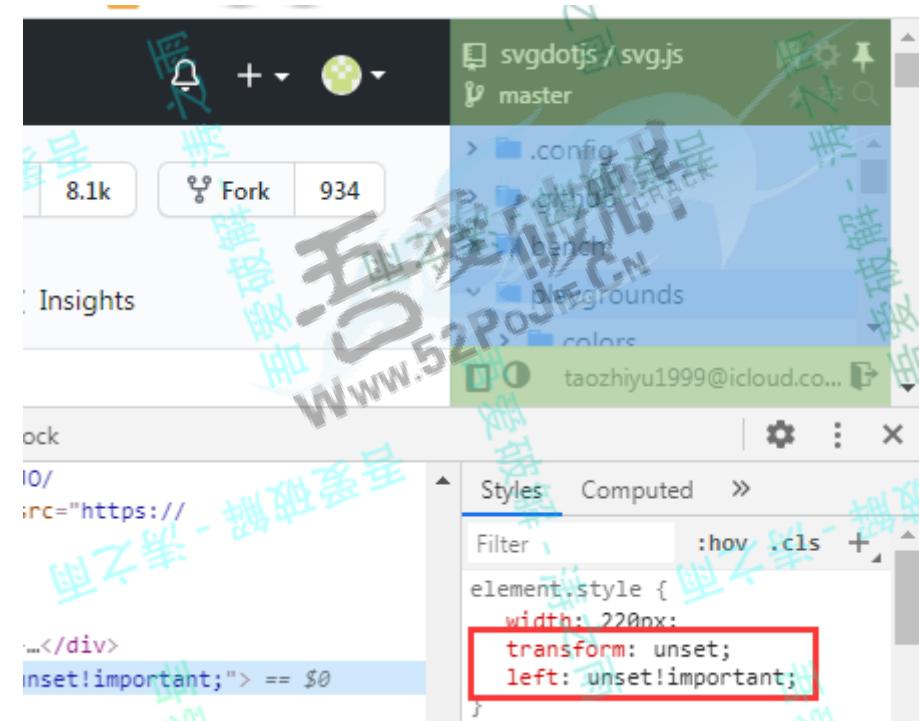
## 给作者填坑

老bug了。。无法滚动的bug修复了，但是界面还是不正常（左侧多余，右侧溢出）



在 Chrome 的控制台里一通摸索(瞎改) ,

一般为 CSS 或 CLASS 的修改居多，需要有 HTML、CSS 等基础  
发现是



把 transform 和 left 属性强制删除，设置就行 ( unset )

在下面的 class 里找到设置右侧属性的类 ( 本例中见下图 )

```
.octotree-dock-right.octotree-show .octotree-sidebar.octotree-github-sidebar {
    transform: translate3d(100vw, 0px, 0px) translateX(-100%);
}

.octotree-dock-right .octotree-sidebar.octotree-github-sidebar {
    transform: translate3d(100vw, 0px, 0px) translateX(0px);
```

因为是 !important 强制属性，所以只需要改一个就行。

修改部分如图所示：

```
ctotree-github-sidebar{-webkit-transform:unset!important;transform:unset!important;left:unset!important}.octotree-dock-right.octotree
ctotree-github-sidebar{-webkit-transform:translate3d(100vw,0,0)+translateX(-100%);transform:translate3d(100vw,0,0)+translateX(-100%)}.octotree
```

注：

1. 改法不唯一，只要能实现目的，且简洁方便快捷实用，就是好方法。
2. 本方法因为没有使用 transform 属性，移动动画会丢失。

不想加 animation 动画，又没想到更好的方法。。。

凑活用吧，反正主要不是看动画的。

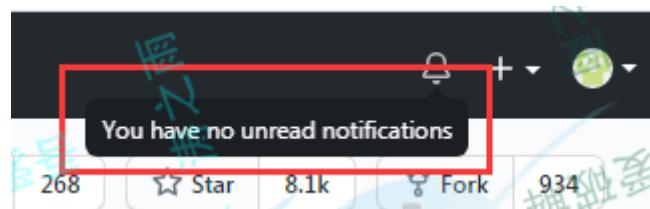
## 强迫症患者



tip 竟然出现在左边。。。

好难受看我改到下 ( 上 ) 面

分析 class , tooltipped-w 是左边显示，因此猜测 w 是指西边。



观察到上面 Git 部分有想要的样式，  
同理炮制出 tooltipped-sw (西南)，改动前后的对比



注：chrome 在标签上右键 Force State 可以设置当前状态

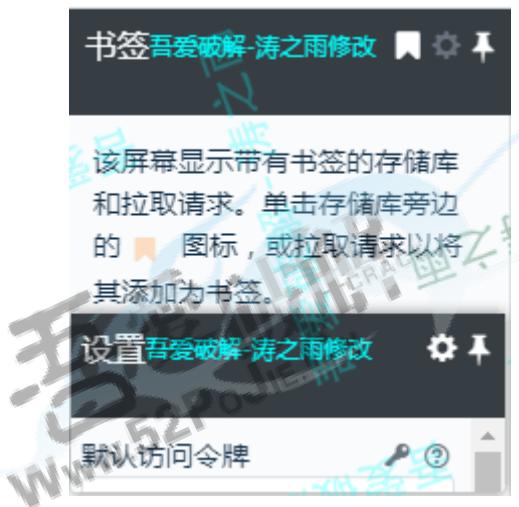
改完代码发现在右侧的时候显示正常，但是左侧的时候就变了。  
查看代码发现因为有页面方向的变换，对 class 进行了动态改变。  
改 js 代码吧。给每一个加上单独的类（这里我使用的是 tao）

```
this._sidebar.find(".tooltipped").each(function() {
  const t = $(this);
  if(t.hasClass("tao")){return;}
  t.hasClass(n.ne) || t.hasClass(n.nw) ? t.toggleClass(n.ne, !e).toggleClass(n.nw, e)
  : (t.hasClass(n.s) || t.hasClass(n.sw)) && t.toggleClass(n.s, !e).toggleClass(n.sw, e)
})
```

效果（注：图经过处理，为了方便看效果，部分间距增加了）



## 其他修改（瞎改）



咳咳，反正 html、css 或 js 呀。

看着改好了。。。

## 修改--杂项

还有一些乱七八糟的东西可以修改，具体可以参考附录中的第二个教程  
比如右键，第一个是官网。

可以点击后弹一个咱们的 哉爱论坛 官网，改改作者、简介啥的

```
{
  "author": "涛之雨", ① 作者名
  "background": {
    "persistent": true,
    "scripts": [ "background.js" ]
  },
  "browser_action": {
    "browser_style": true,
    "default_icon": "icons/icon128.png"
  },
  "content_scripts": [ {
    "css": [ "content.css", "themes-0.css", "themes-1.css", "themes-2.css", "themes-3.css" ],
    "js": [ "content.js" ],
    "matches": [ "https://github.com/*" ],
    "run_at": "document_start"
  }],
  "description": "吾爱破解--涛之雨修改, GitHub插件", ② 描述
  "homepage_url": "https://www.52pojie.cn", ③ 右键第一个官网的链接
  "icons": {
    "128": "icons/icon128.png"
  },
  "key": "MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIABCgKCAQEAngvmj3AHZqcpXeKfJYYI7ETsWIQQBGwkytHIrI6dsDun
    /begZqxckSLjn1S+wG8MP5CM1HxspnCVyaoTB9Nis7C6V7GYrwLlx6v4wM1ET4vArvhgGLKsTpM0q1FvE18013ViGa9zb
  X4U9DdVl1rpa2xs ④ 签名，可以删掉（之后打包的时候会自动生成）这是就是我自己生成的) u4Sa7
  WekiEYqlyFKtwPe1fM9Mu24RLkkktisiwfBZD2Y4FAvCCwsuqI3G8rsEw77JbSaH3+J+U0taiJz+o6paa9boyJPwem/
    pRQew161531HVZ1LQIDAQAB",
  "manifest_version": 2,
  "minimum_chrome_version": "60",
  "name": "Octotree",
  "optional_permissions": [ "http:///*", "https:///*", "webNavigation", "tabs" ],
  "permissions": [ "https://api.github.com/*", "https://www.octotree.id/*", "storage", "contextMenus", "activeTab" ],
  "update_url": "", ⑤ 更新链接。。。我反正不让他更新。当然如果有需要可以自建服务器
  "version": "5.2.1",
  "web_accessible_resources": [ "*.woff2", "*.png", "*.gif", "*.svg" ]
}
  (仅做演示)，保存的时候记得删掉
```

注！上述截图因为方便说名，仅把更新链接删掉了。真正操作的时候请删除

或是卸载后再弹一下 5 爱的官网（下图第一个框）

当然也可以像我上个版本一样，第一次安装的时候弹一下。

```
(function() {
  'use strict';
  var message, messageStatus, statuscode;
  if (chrome.runtime.lastError) {
    message = chrome.runtime.lastError.message;
    messageStatus = chrome.runtime.lastError.status;
    statuscode = chrome.runtime.lastError.code;
  }
  eventNames = return_sendRequest(o,e).listen();
  chrome.runtime.setUninstallURL("https://www.52pojie.cn");
  function handleInstalled(details) {
    if (details.reason === "install") {
      alert("感谢安装，\n因为一些特殊原因，本插件在不科学的网络环境下，部分网页无法加载。。。\\n\n最好用正确的姿势上网。。。\\nCracked By：吾爱破解_涛之雨");
      chrome.tabs.create({url: "https://www.52pojie.cn"});
    }
    chrome.runtime.onInstalled.addListener(handleInstalled);
  }
})()
```

效果：

第一次安装有弹窗：



同时点击确定后有吾爱网站弹出



## 修改--动态调试

动态调试是最方便找到出问题的位置以及原因的办法（虽然不一定能想到怎么解决。比如下图）。



亦或是代码被混淆的很厉害，又抽取了结构，平流抽象化了（AST）

注：AST又被称为js预编译，可以参考[这里的介绍](#)

还是以 octotree 为例

假如我故意（一不小心）出了一个未知错误。

! Uncaught (in promise) Error: Extension context invalidated.

上下文

<https://github.com/svgdotjs/svg.js/blob/master/bench/runner.html>

堆栈追踪

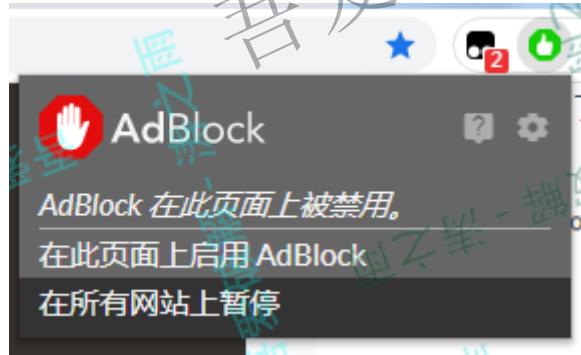
```
content.js:1 (匿名函数)
content.js:1 (匿名函数)
content.js:1 (_innerGet)
content.js:1 (get)
content.js:2 (_cacheTabsMetadata)
content.js:2 (_setTabMetadata)
content.js:2 (_saveCurrentTabState)
content.js:1 (dispatch)
content.js:1 (g.handle)
```

插件的错误也十分“委婉”，而且因为压缩了，不方便看错误位置。  
动态调试吧。

这里要分情况来看。

## popup ( 弹出页面 )

就是点击插件的按钮，会有窗口弹出的。类似于下图的 AdBlock 插件

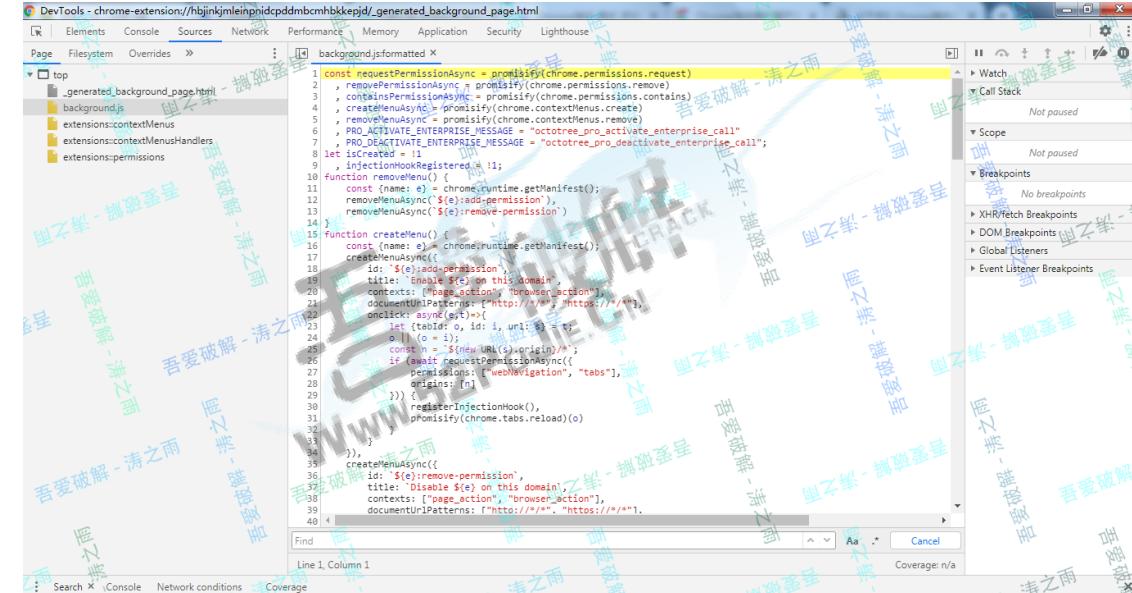


插件上 右键 - 审查弹出窗口 - (格式化) 下断点 - 重新加载

注：重新加载可以在 console 里输入

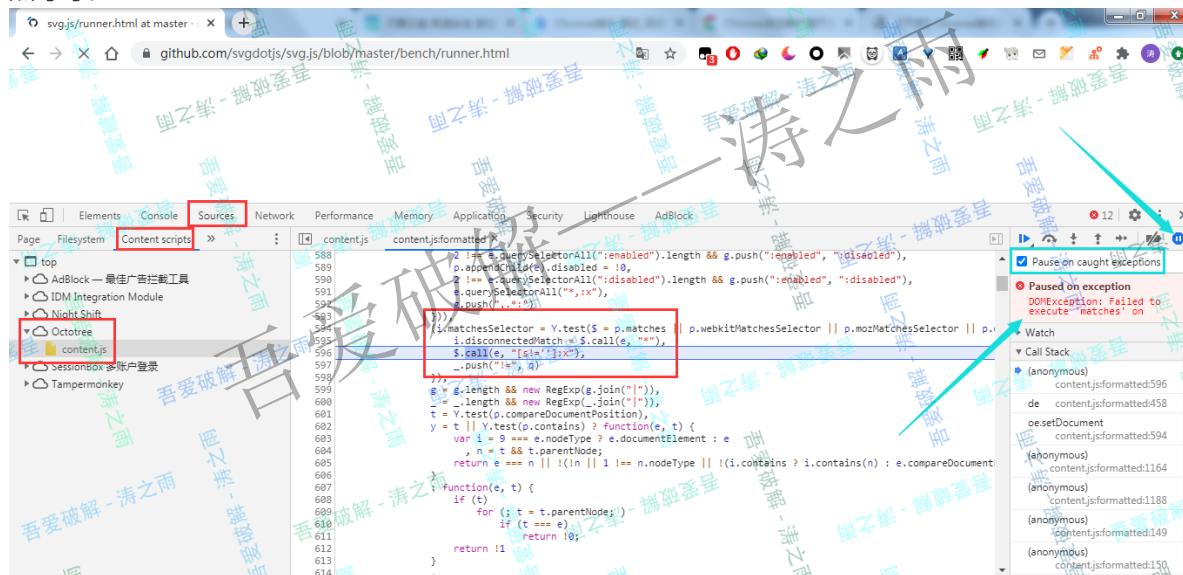
```
1 | location.reload(true)
```

## 重新加载。



## 植入到页面的js

可以直接在 `console - Content Script -` 页面的位置（如果有`iframe`的话）- 插件名 中格式化，下断点等等。



注：推荐一个我刚发现的功能（可能有朋友早就知道了了吧。。）

在上图蓝色箭头所示的方框的位置，是异常前下断点，

系统会在出现异常并且没有被捕获的地方，在报错前断下。堆栈，数据都保存着，而且有报错的信息，可以方便调试。

## 出现错误的处理方法



最简单的办法就是：

一直撤销到不报错为止，然后重做。  
如果不行的话，请退回到备份版本。

如果重复好几次，还是感觉没问题，就需要动态调试了（参上文）。

## 打包

该改的都改完了，该填的坑也都填好了。

需要打包发布了（如果有账号的话）

点击上方的“打包扩展程序”，输入扩展所在的文件夹

点击 打包扩展程序 即可。

至于 私钥 如果你不是第一次打包，上一次打包后会自动生成一个私钥。你可以用这个私钥重新打包（本质就类似安卓的签名）



补充：

1. 每个私钥都标志这一个独立的插件，如果不同的插件用同一个私钥签名，会导致覆盖。  
(惨痛教训)
2. 因为Chrome的特性，无法直接安装没有在谷歌商店中发行的插件（除非签名经过官方许可，并且有第三方进行安装，比如下面的区别示意图）



如果是分享 zip 版未打包的插件，推荐先打包后再重命名。  
这样可以保证插件数据不会丢失  
更新的时候，因为有 key 系统会认为是在更新插件，不会导致多个版本共存  
( 5555 我才知道，所以现在。。。 )



新版本旧版本共存了。。。也就是说旧版本的设置需要手动转移到新版本 ( 气哭 )

## 参考资料

有问题或是遇到奇怪的 API 可以查阅。

内容	链接	作用
GitHub翻译版	<a href="#">点我</a>	咳咳，不一定是最新的。 至少比英文原版方便吧
某网站分享的经验和教程	<a href="#">点我</a>	偏实战吧，比较详细 ( 虽然是转的，但是原作者图片都没了 )
图灵的图书版	<a href="#">点我</a>	emmm没啥特别的
官方教程	<a href="#">点我</a>	最权威的官方教程 ( 纯英语、正确的上网 方式警告 )
360汉化版(超级旧的版本)	<a href="#">点我</a>	旧版本，很多图都挂了

## 修改样本

原 crx

Chrome 插件下载地址 ( 谷歌商店 ) : [链接太长。。。点我访问](#)

下载 :

<https://cloud.189.cn/t/r2MJ3yaMVV3e> ( 访问码 : 03ts )

注：后缀改成zip即可

【原包】即为未修改版本，【样品】即为修改后的版本