Chrome 扩展开发指南(1)——入门

这是制作 chrome 扩展插件的入门指南，不需要任何编程基础，看完这个后，我们就着手做自己的 Chrome 插件了。好吧，我们现在就开始，其实我也是个新手。

准备工具

做任何事情都要有个工具，制作 chrome 插件需要的工具很少。

- 记事本，用来编写代码
- Chrome 浏览器，这个不能少吧。Windows 下，所有版本的 Chrome 都可以制作插件。Linux 下需要下载 Beta 版本，Mac 下载 dev 版本。

开始制作第一个插件

- 在计算机中创建一个目录来存放插件代码。
- 在目录里面创建文件 manifest.json(注意后缀名是.json)，用记事本打开，写入如下代码

```
1  {
1    "name": "第一个 Chrome 插件",
1    "version": "1.0",
1    "description": "我的第一个 Chrome 插件，还不错吧",
1    "browser_action": {
1      "default_icon": "icon.gif"
1    }
1  }
```

复制代码

- 把下面两张图片保存到文件夹中，分别取名 icon.gif 和 smile.gif

图片一：　　图片二：

- 安装这个插件：

a.点击右上角扳手，选择扩展程序，打开扩展中心。

b.点击右上角的“开发人员模式”，使得前面的“+”变成“-”，打开相应的菜单。如果一开始就是“-”，那么不用点击。

c.点击“载入正在开发的扩展程序按钮”，导入刚才创建的文件夹。

如果一切顺利，你的 Chrome 地址栏将会有个新图标，你的第一个插件诞生了。

给第一个插件增加新功能

你现在虽然做了第一个插件，但实际上他并没有实现任何功能，我们点击图标，没有任何反应。下面我们就给他增加点功能。

- 编辑 manifest.json 这个文件，用下面的代码替换现有的代码，其实我们只是加了一行代码和一个逗号而已。

```
1  {
1    "name": "第一个 Chrome 插件",
1    "version": "1.0",
1    "description": "我的第一个 Chrome 插件，还不错吧",
1    "browser_action": {
1      "default_icon": "icon.gif",
1      "popup": "popup.html"
1    }
1  }
```

复制代码

- 下面我们创建一个文本文件 popup.html，用记事本打开，将下面的代码写进去

```
1  <p>Hello,Chrome!</p>
1  <p>我的名字叫 ChromeChina!</p>
1  <p><a href="http://www.chromechina.com" target="_blank">Chrome 中文论坛欢迎你
```

```
</a>
```

```
1    <p><img src="smile.gif" /></p>
```

复制代码

- 回到 Chrome 的扩展中心，点击插件下的“重新载入 ”。

现在点击插件图标看看。我们的第一个插件算是制作成功了。

打包插件

我们想把自己制作的插件给其他人用，那么就需要将插件打包。

- 回到 Chrome 的插件扩展中心，点击“打包扩展程序”按钮。
- 选择刚才创建的文件夹，点击确定生成后缀为 crx 和 cpm 文件各一个。

把 crx 文件发送给自己的朋友，告诉他们你也会制作 chrome 插件吧。

你可以修改里面的文字图片，制作出极具个性的扩展来了。现在让我们来看看大家都做出了什么样的扩展，把你的第一个扩展上传上来让大家看看吧！这个是我的：

Chrome 扩展开发指南(2)——概述

这篇文章翻译自 http://code.google.com/chrome/extensions/overview.html，我还没有真正做过插件，翻译这篇文章算是班门弄斧，有翻译的不好的地方请一定指出来，希望能够达到完美！

(对新手说的话：文章涉及到的一些术语，对没有任何网页知识的新手来说还是挺难懂的，可以借助 Google/baidu 看一下，其实不懂也没多大关系，这篇文章只是一个概述，完全可以跳过这些术语，希望大家对制作插件不要失去信心。)

只要看完这篇文章，并且做过入门指南中的例子，你就可以真正开始开发属于自己的 Chrome 插件了。

基础知识

一个 Chrome 扩展是由 HTML、CSS、JavaScript、图片等文件压缩而成。扩展实际上就是一个 web 页面，你可以用任何浏览器提供给 web 页面的接口，从 XMLHttpRequest 到 JSON ，再到 HTML 本地缓存都可以使用。

Chrome 扩展能做什么呢？我们肯定使用过一些扩展，会发现有些扩展在 Chrome 地址栏右侧

区域增加一个图标。还有些扩展能够和浏览器的一些元素(如书签、tab 导航标签)交互。扩展还可以和 web 页面交互，甚至是从 web 服务器获取数据。更加详细的内容可以从 Developer's Guide 看到。

Chrome 扩展的组成文件

每个扩展由下列文件组成：

- 一个 manifest 文件(主文件,json 格式)
- 至少一个 HTML 文件(主题可以没有 HTML 文件)
- JavaScript 文件 (可选，非必须)
- 任何其他你需要的文件(比如图片)

当你开发一个扩展的时候，需要把这些文件放在一个文件夹里，当你发布这个扩展的时候，这个文件夹下的所有文件将会打包成一个特殊后缀.crx 的 ZIP 文件。

引用文件

你可以放置任何文件到你的扩展里面，但是怎么调用这些文件呢？一般来说，使用相对地址调用，类似 HTML 中调用文件。下面是个例子，在子文件夹 images 中有个图片 myimage.png，我们可以这样调用它

```
1    <img src="images/myimage.png">
```

复制代码

其中 images/myimage.png 表示这个文件。

也许你注意到当使用 Google Chrome debugger 查看这些文件的时候给，每个文件的地址是下面这种格式

```
2    chrome-extension://<extensionID>/<pathToFile>
```

复制代码

这个地址中，<extensionID>是你制作的扩展的唯一标示符，也就是扩展的身份证编号。<pathToFile>是文件相对扩展顶级文件夹得位置。

manifest 文件

主文件取名 manifest.json，用来描述这个扩展，包括扩展名字、版本、调用的文件、可用域等信息。下面是个典型的 manifest 文件，这个扩展可以调用 google.com 的内容。

```
3  {
4      "name": "My Extension",
5      "version": "2.1",
6      "description": "Gets information from Google.",
7      "icons": { "128": "icon_128.png" },
8      "background_page": "bg.html",
9      "permissions": ["http://*.google.com/", "https://*.google.com/"],
10     "browser_action": {
11       "default_title": "",
12       "default_icon": "icon_19.png",
13       "popup": "popup.html"
14     }
15 }
```

复制代码

扩展结构组成结构

绝大部分扩展有 background 文件，一个不可见的文件控制着整个扩展的运行。

上面这个图片显示的浏览器至少安装了两个扩展：一个浏览器行为扩展(黄色的图标),页面行为扩展(蓝色的图标)。这个浏览器行为扩展的 background 文件是用一个 HTML 文件定义的(background.html)，这个 background 文件中有 JavaScript 代码控制整个浏览器的活动。

HTML 页面

background 不是唯一存在的 HTML 文件，比如浏览器行为可能是弹出一个小窗口，这个小窗口的内容就可以调用一个 HTML 文件。Chrome 扩展也能够用 chrome.tabs.create() or window.open()这种函数来显示 HTML 文件。

扩展里面的 HTML 文件可以互相访问对方的 DOM 结构，可以引用其他文件中定义的函数。

下面的图展示了浏览器弹出一个窗口这个功能的结构(这正是我们最开始的例子)。这个弹出窗口的内容是一个 HTML 的 web 文件，这个弹出窗口不需要包含 background 文件中的代码，因为，popup.html 和 background 是可以互相访问的。

内容脚本(Content scripts)

如果你插件需要和网页交互，那么他就需要一个内容脚本(Content scripts)，内容脚本常由 JavaScript 编写，会在网页载入完成后调用。完全可以把内容脚本看做是网页的一部分，而不是扩展的一部分。

内容脚本可以访问到当前浏览器浏览的页面，而且还可以改变网页的显示方式(油猴脚本就是内容脚本)。下面的图片中，内容脚本可以读取、更改网页的 DOM。注意，他不能更改 background.html 中的内容。

内容脚本也不是和父扩展完全隔离开来，他也可以和父级扩展交换信息。如下图中所示，内容脚本在发现一个 RSS Feed 地址后将会给 background.html 发送一个信息。或者 background.html 给内容脚本发送一个信息要求改变网页外观。

不同页面间的交互

一个扩展中的文件常常需要交互。由于扩展的所有文件都由同一个进程执行，网页能够直接给其他页面发送命令。

可以使用类似 chrome.extension methods such as getViews() and getBackgroundPage()这样的方法引用扩展中的方法。一旦页面中引用了另外的页面，第一个页面就可以调用其他页面的函数，甚至可以控制 DOM。

结束语

好了，你已经大概了解了一个扩展程序的基本内容，可以开始写作自己的扩展了。

Chrome 扩展开发指南(3)——Browser Action(扩展图标)

这是扩展开发指南的第三篇，前面我们首先作了第一个扩展，然后学习了 Chrome 扩展的大概结构，看完后可能会有些迷惑，别担心，相信随着我们学习的深入，我们渐渐发现我们已经可以做扩展了。当然为了做出优秀的扩展，我们还需要学习一些 HTML、CSS、JavaScript 的基础知识，http://www.w3school.com.cn/网站就不错。

今天的文章翻译自 http://code.google.com/chrome/extensions/browserAction.html，介绍 Browser Action，即右侧的扩展图标。这节的内容还是挺有趣的。(同样，有翻译需要改进的地方请指出来)

Browser Actions 的作用就是控制 Chrome 地址栏右侧添加一个图标。除了给 chrome 增加一个图标的功能外，还可以设置提示文字、图标标记、弹出窗口。

下图中，在地址栏右侧的彩色图标就是一个 Browser Action。

Browser Actions 创建的图标是一直可见的，如果你想创建一个不是一直不可见的图标，可以使用 page action。

Browser Action 在 Manifest 文件中的位置

下面是个在扩展的 manifest 文件中注册 browser action 的例子：

```
{
  "name": "My extension",
  ...
  "browser_action": {
    "default_icon": "images/icon19.png", // required
    "default_title": "Google Mail",      // optional; shown in tooltip
    "default_popup": "popup.html"        // optional
  },
  ...
```

```
10 }
```

复制代码

UI 部分

Browser Action 必须有一个图标。同时还可以有提示文字、图标标记、弹窗。

图标

Browser Action 的图标会被浏览器缩放成19px*19px 大小，太大的图标是没有意义的。

你可以用两种方法定义图标：用一个静态图片，或者用 HTML 中的 canvas 元素。用静态图片的话简单些，但是用 canvas 元素可以创建更加平滑的图片。

静态图片可以是任意常见格式的图片，包括 BMP, GIF, ICO, JPEG, or PNG。

我们可以在 manifest 文件中用 default_icon 语句来定义这个图标，也可以调用 setIcon()函数。

提示文字

提示文字是指将鼠标移到扩展图标上显示的文字。我们可以在 manifest 中用 default_title 定义，也可以通过调用 setTitle()函数。

图标标记

图标标记是指覆盖在扩展图标上的一些文字，比如 Gmail 提醒图标上未读邮件数，PR 查询工具上 PR 值。由于标记的位置很小，他最多只能容纳4个字母。

设置标记文字或者背景可以分别使用 setBadgeText() and setBadgeBackgroundColor()。

弹窗

当我们点击一些扩展的时候，会发现有个小弹窗出现,比如我们一开始的例子中。这个弹窗可以包含任何 HTML 内容，他的大小也是和内容自适应的。

给 Browser Action 增加弹窗可以在 manifest 的 default_popup 定义弹窗中显示的 html 文件名字，当然也可以使用 setPopup()函数。

几个小提醒

为了扩展更加美观，请遵守下列守则：

仅在这个扩展需要在大部分页面运行的时候才使用 browser action

仅在小部分页面起作用的话就不要用 browser action，而是用 page actions。

使用显眼的图标

不要试图模仿 chrome 浏览器原有的扳手/页面图标，你的扩展要独特一些。

你的图标边缘应该使用 alpha 透明，这样的话可以融合到各种不同的浏览器主题里。

例子解析

激动人心的时候来了，在这个文件夹下 examples/api/browserAction 有些 browser action 的例子。其中有个 set_page_color，我们试着重新编写他。

首先我们知道，首先新建一个文件夹 myExtension 用来存放所有文件，我们知道每个 Chrome 扩展需要有个 manifest.json 文件来描述这个扩展，新建文件 manifest.json，用文本编辑器打开，输入：

```
{
    "name": "我的扩展实例",
    "version": "1.0",
    "browser_action": {
        "default_title": "Set this page's color.",
        "default_icon": "icon.png",
        "popup": "popup.html"
    }
}
```

复制代码

这是一个很简单的 manifest.json 文件模板，其中 browser_action 就是这篇文章降到的东西，default_title 是描述，default_icon 是图标，popup 是弹窗。这里的弹窗调用了 popup.html 文

件，我们再创建一个文件 popup.html，popup.html 是个普通的 HTML 文件，内容如下：

```
<style>
body {
    overflow: hidden;
    margin: 0px;
    padding: 0px;
    background: white;
}

div:first-child {
    margin-top: 0px;
}

div {
    cursor: pointer;
    text-align: center;
    padding: 1px 3px;
    font-family: sans-serif;
    font-size: 0.8em;
    width: 100px;
    margin-top: 1px;
    background: #cccccc;
}
```

```
42  div:hover {
43      background: #aaaaaa;
44  }
45  #red {
46      border: 1px solid red;
47      color: red;
48  }
49  #blue {
50      border: 1px solid blue;
51      color: blue;
52  }
53  #green {
54      border: 1px solid green;
55      color: green;
56  }
57  #yellow {
58      border: 1px solid yellow;
59      color: yellow;
60  }
61  </style>
62  <script>
63  function click(color) {
```

```
64      chrome.tabs.executeScript(null,
65          {code:"document.body.style.backgroundColor='" + color.id + "'"});
66      window.close();
67  }
68  </script>
69  <div onclick="click(this)" id="red">red</div>
70  <div onclick="click(this)" id="blue">blue</div>
71  <div onclick="click(this)" id="green">green</div>
72  <div onclick="click(this)" id="yellow">yellow</div>
```

复制代码

这个文件的内容有三种语言，HTML、CSS、JavaScript，这三种语言组成一个基本的网页，如果你还不是很清楚的话可以以后慢慢学些。其中调用了 Chrome 接口函数 chrome.tabs.executeScript，也是以后会看到的。整个文件的意思是：1、显示四格不同颜色的矩形框，2、当点击这些矩形框的时候变换页面背景色。

我们还需要一个图标显示在工具栏上，把这个图片保存到文件夹中　　。

好了，我们的扩展制作完成了，载入他们测试一下吧！

如果有一些 JavaScript 知识，可以修改这些扩展，创建一些丰富多彩的效果。

比如把 popup.html 中的

```
73  function click(color) {
74      chrome.tabs.executeScript(null,
75          {code:"document.body.style.backgroundColor='" + color.id + "'"});
76      window.close();
77  }
78  </script>
```

```
79  <div onclick="click(this)" id="red">red</div>
80  <div onclick="click(this)" id="blue">blue</div>
81  <div onclick="click(this)" id="green">green</div>
82  <div onclick="click(this)" id="yellow">yellow</div>
```

复制代码

换成

```
83  function click(color) {
84      chrome.tabs.executeScript(null,
85              {code:"document.getElementById('lg').getElementsByTagName('img')[0].src='" +
color.title + "'});
86      window.close();
87  }
88  </script>
89  <div onclick="click(this)" id="red"
title="http://www.google.com.hk/intl/zh-CN/images/logo_cn.gif">Google</div>
90  <div onclick="click(this)" id="blue">blue</div>
91  <div onclick="click(this)" id="green">green</div>
92  <div onclick="click(this)" id="yellow">yellow</div>
```

复制代码

在百度主页上打开这个扩展，点击第一个按钮"Google"，可以把百度的 logo 换成 google 的。

Chrome 扩展开发指南(4)——Options Pages（选项页面）

你可以提供一个选项页面(Options Pages)让用户自定义你的扩展。如果设置了选项页面，那么

扩展管理页 chrome://extensions 将会有一个链接指向选项页面。

定义选项页面包括两步：

1、在 manifest 中定义选项页

```
{
    "name": "My extension",
    ...
    "options_page": "options.html",
    ...
}
```

复制代码

上例中，options_page 代表选项页面，options.html 是具体的文件地址。

2、编写选项页面

选项页面是一个典型的网页，下面是一个选项页面的例子：

```
<html>
<head><title>My Test Extension Options</title></head>
<script type="text/javascript">

// Saves options to localStorage.
function save_options() {
    var select = document.getElementById("color");
    var color = select.children[select.selectedIndex].value;
    localStorage["favorite_color"] = color;

    // Update status to let user know options were saved.
```

```
18    var status = document.getElementById("status");
19    status.innerHTML = "Options Saved.";
20    setTimeout(function() {
21      status.innerHTML = "";
22    }, 750);
23  }
24
25  // Restores select box state to saved value from localStorage.
26  function restore_options() {
27    var favorite = localStorage["favorite_color"];
28    if (!favorite) {
29      return;
30    }
31    var select = document.getElementById("color");
32    for (var i = 0; i < select.children.length; i++) {
33      var child = select.children[i];
34      if (child.value == favorite) {
35        child.selected = "true";
36        break;
37      }
38    }
39  }
40
```

```
41 </script>
42
43 <body onload="restore_options()">
44
45 Favorite Color:
46 <select id="color">
47 <option value="red">red</option>
48 <option value="green">green</option>
49 <option value="blue">blue</option>
50 <option value="yellow">yellow</option>
51 </select>
52
53 <br>
54 <button onclick="save_options()">Save</button>
55 </body>
56 </html>
```

复制代码

注意事项

早期版本的 chrome 可能不支持这个功能。

我们正计划提供一个默认的 css 来使得不同扩展的选项页面保持风格一致，你可以从这里(http://crbug.com/25317)查看最新的进展。

知识补充

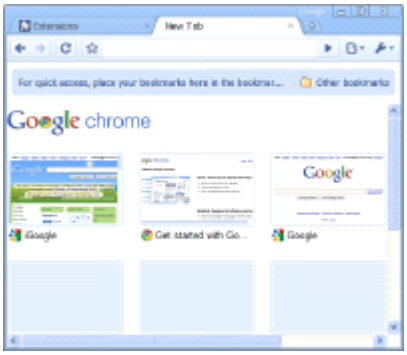
上面的例子中使用 LOCALSTORAGE 保存数据，具体介绍可以查看《使用 LOCALSTORAGE 保存数据》

Chrome 扩展开发指南(5)——Override Pages(重置页面)

重置是一种用自己提供的页面替换 Google Chrome 默认页面的方法。一个重置页面常常是用 HTML、JavaScript、CSS 组成。

当前，能够替换的页面只有新标新标签页，新标签页就是我们打开一个新标签时出现的页面。

我们可以把默认的新标签页：



替换成这种样式：

重置页面非常简单，只需在 Manifest 中定义自己的页面地址。比如下面的例子中，我们使用了 newtab.html 来重定义新标签页。

```
1  {
2      "name": "My extension",
3      ...
4      "chrome_url_overrides": {
5          "newtab": "newtab.html"
```

```
6       },
7       ...
8   }
```

复制代码

几点注意事项

为了让你定义的新标签页看起来不错，请遵循下面几点建议：

- **保持页面简洁，使得能够快速加载**

由于新标签页经常出现，外观就显得特别重要。比如我们要避免从远程调用数据，或者读取数据库资源。

- **确保有<title>标签**

如果没有<title>，大家讲会看到页面的 URL，这会让人很迷惑，我们应该包含这样一句 <title>New Tab's Name</title>

- **不要让键盘焦点在页面上**

我们应该让用户新建标签页的时候键盘焦点在地址栏上。

- **不要模仿默认的新标签页面**

创建默认标签页的 API（比如最近关闭的标签、最常访问的网站等等）不存在！你必须做出一些完全不同的东西。

例子

这儿 examples/api/override 有一些重置新标签页的例子。

其中有个我们至学习以来碰到的最简单的例子，把新标签页面换成空白页面

新建 manifest.json 文件：

```
9   {
10      "name": "空白的新标签页",
11      "version": "0.1",
12      "chrome_url_overrides": {
13          "newtab": "blank.html"
```

```
14      }
15  }
```

复制代码

新建文件 blank.html 作为默认标签页，我们可以只写这样一句话：

```
16  <title>新标签页</title>
```

复制代码

好的，看看效果吧，就这么简单，你现在就可以动手 DIY 了。

原文 http://code.google.com/chrome/extensions/override.html

由 ChromeChina 翻译，转载注明出处 http://dev.chromechina.com/

Chrome 扩展开发指南(6)——Page Actions(地址栏图标)

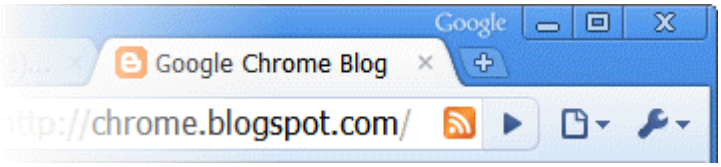
地址栏图标代表的是页面内功能，这种功能只能在特地的当前页面发生，而不是所有页面都有这种功能，也就是并不是所有页面上都能显示这样一个图标。

比如以下功能只在特定页面才有：

RSS 订阅图标(仅在有 rss 订阅功能的网页中才显示)

幻灯片显示照片（仅在特定的照片分享网页中有这种功能）

下面是个显示 RSS 订阅图标的图：



如果你希望能够一直显示这个图标，可以使用扩展图标。

Manifest

定义地址栏图标使用类似下面的 manifest 文件：

```
1   {
2       "name": "My extension",
3       ...
```

```
4       "page_action": {
5         "default_icon": "icons/foo.png", // required
6         "default_title": "Do action",       // optional; shown in tooltip
7         "default_popup": "popup.html"       // optional
8       },
9       ...
10  }
```

复制代码

其中 page_action 代表地址栏图标相关参数，也就是页面功能的参数。

图标部分

类似于扩展图标，地址栏图标需要一个图片，同时也可以有提示和弹出窗口。但是他们不能带有徽标，但是地址栏图标能够显示/隐藏。有关提示和弹出窗口的更多谢谢可以查看扩展图标。

我们可以用 show()和 hide() 函数分别使地址栏图标显示和隐藏。默认情况下，地址栏图标是隐藏的。当你要显示这个图标的时候，你必须指定哪一个 tab 页面可以显示，这时候只有关闭了标签页或者打开了一个不一样的网页地址地址栏图标才会不显示。

几点注意

请遵守以下几点约定：

- 在一部分页面中起作用的功能才使用地址栏图标
- 大部分页面起作用的功能请使用扩展图标。
- 请使用比扩展图标更小的图标，一般来说地址栏图标都小于19px，并且边缘会有模糊效果。
- 不要总是激活你的图标，那是很讨厌的。

实例

你可以在 examples/api/pageAction 文件夹下找到一些例子。

原文 http://code.google.com/chrome/extensions/override.html

由 ChromeChina 翻译，转载注明出处 http://dev.chromechina.com/

Chrome 扩展开发指南(7)——Themes(主题制作)

在 Chrome 中，主题当做一个特殊的插件处理。主题要像普通文件一样打包，但是主题中不包含 HTML 或者 JavaScript 代码。

你可以在主题库里找到一些主题下载。

下面只是简单介绍下主题制作，详细的内容可以参考

http://code.google.com/p/chromium/wiki/ThemeCreationGuide

主题是在 Manifest 文件中定义的。下面是个定义主题的典型例子：

```
1  {
2    "version": "2.6",
3    "name": "camo theme",
4    "theme": {
5      "images" : {
6        "theme_frame" : "images/theme_frame_camo.png",
7        "theme_frame_overlay" : "images/theme_frame_stripe.png",
8        "theme_toolbar" : "images/theme_toolbar_camo.png",
9        "theme_ntp_background" : "images/theme_ntp_background_norepeat.png",
10       "theme_ntp_attribution" : "images/attribution.png"
11     },
12     "colors" : {
13       "frame" : [71, 105, 91],
14       "toolbar" : [207, 221, 192],
```

```
15            "ntp_text" : [20, 40, 0],
16            "ntp_link" : [36, 70, 0],
17            "ntp_section" : [207, 221, 192],
18            "button_background" : [255, 255, 255]
19        },
20        "tints" : {
21            "buttons" : [0.33, 0.5, 0.47]
22        },
23        "properties" : {
24            "ntp_background_alignment" : "bottom"
25        }
26    }
27 }
```

复制代码

我们看到在 theme 类下有几个元素，分为为 images、colors、tints、properties。下面分别介绍他们。

colors

用来定义基本颜色。颜色需要用 RGB 格式表示，你可以在 browser_theme_provider.cc 查看到底可以定义哪些内容。

images

图片需要用相对地址引用，你设置 browser_theme_provider.cc 文件中 kThemeableImages 数组的所有元素。去掉 IDR_并且转化成小写格式后就是你需要设置的东西，比如 IDR_THEME_NTP_BACKGROUND 需要转化为 theme_ntp_background。

properties

这个地方用来定义诸如背景定位方式、背景重复等属性。browser_theme_provider.cc 里面可以看到有哪些属性可以定义。

tints

你可以给部分 UI 着色，比如按钮、框架、背景 tab 标签。(这里翻译可能有问题，chromechina 注)

原文 http://code.google.com/chrome/extensions/themes.html 由 ChromeChina 翻译，转载注明出处 http://dev.chromechina.com/

Chrome 扩展开发指南(8)——Bookmarks(书签操作)

我们可以用 chrome.bookmarks 模块来对书签做创建、组织等操作。

Manifest

要对书签进行操作，必须要再 Manifest 文件中设置允许调用书签接口(bookmarks API)。一般像下面这样写：

```
1  {
2      "name": "My extension",
3      ...
4      "permissions": [
5        "bookmarks"
6      ],
```

```
7       ...
8   }
```

复制代码

书签对象和属性

书签以树形机构组织，节点或是一个书签或者是个文件夹(可以包含多个书签)。每个节点都是一个 BookmarkTreeNode 对象。BookmarkTreeNode 对象在接口中会常常用到。比如我们调用 create()来创建书签的时候，我们要传递一个这个新节点的父节点(parentId 属性)，另外还可以传递 index(兄弟节点中的排序，0开始), title(标题), url(地址)。

例子

下面代码调用 create()创建了一个书签文件夹取名“Extension bookmarks”，第一个参数是个 json 格式的对象，第二个参数定义了一个函数，这个函数将在创建完书签后调用。

```
9   chrome.bookmarks.create({'parentId': bookmarkBar.id,
10                           'title': 'Extension bookmarks'},
11                          function(newFolder) {
12    console.log("added folder: " + newFolder.title);
13  });
```

复制代码

下面的例子里创建了一个指向这份开发文档的书签。这代码里面没有定义回调函数(callback function).

```
14  chrome.bookmarks.create({'parentId': extensionsFolderId,
15                           'title': 'Extensions doc',
16                           'url': 'http://code.google.com/chrome/extensions'});
```

复制代码

更多操作书签的例子可见在 basic bookmarks sample 找到。

有关 BookmarkTreeNode 对象的完整定义以及所有的书签函数可以点击这里查看，这部分的内容计划放到最后翻译。

本文翻译自 http://code.google.com/chrome/extensions/bookmarks.html

由 ChromeChina 翻译，欢迎到 dev.chromechina.com 交流 Chrome 扩展开发技术。

Chrome 扩展开发指南(9)——Events(事件)

事件(Event)就是当某些行为发生后要触发的函数。这些行为包括打开了一个新标签、点击了一个按钮等。下面是利用 chrome.tabs.onCreated 行为(创建一个新标签页)，只要打开了一个新标签页相应的事件就会被触发。

这个例子中，我们用 addListener()来注册函数，addListener()的参数总是一个函数，但是这个函数的参数根据事件类型不一样而不同。点击这里查看 chrome.tabs.onCreated 的更多信息，你会发现这个函数有一个参数 tab：一个用来描述新标签页的 Tab 对象。

下面的三个函数在任何事件中都可以使用：

```
1  void addListener(function callback(...))
2  void removeListener(function callback(...))
3  bool hasListener(function callback(...))
```

复制代码

本文翻译自 http://code.google.com/chrome/extensions/events.html

由 ChromeChina 翻译，欢迎到 dev.chromechina.com 交流 Chrome 扩展开发技术。