Google chrome extensions

An Introduction to Development

Will Webberley

Overview

- The increasing importance of extensions
- General introduction and examples
- Implementation how-to
- Demonstrations
- Chrome API

Why use them?

- Web applications becoming more important (e.g. Chrome OS)
- Increase productivity
- Generally quick and easy to install and use
- Provide much more functionality to the browser
- Automatically cross-platform and are synchronised
- Chrome is the fastest-growing (in popularity) browser

What do they do?

- Add new CSS to the page
- Provide new functionality through JavaScript
- Read and interact with the page's DOM
- Create new hyperlinks
- Override Chrome's pages (new tab, history pages, etc.)
- Desktop notifications
- ... and many more.

Examples

- Facebook photo-enlarger
- Ad-blockers
- Chrome to Phone
- Grooveshark
- Reddit Enchancement Suite

How do you access them?

- From the "wrench" menu
- Many websites offer extension galleries
- Individual websites/services provide individual extensions
- Google Chrome web store (chrome.google.com/webstore)
- Develop your own!

Development: Basics

- Quick to develop - no SDK or libraries necessary

- You only really *need* 2 files
  - manifest file (JSON)
  - some form of HTML (typically containing CSS and/or JavaScript)

- Further files add more functionality
  - icons (to make it look pretty and for the web store)
  - other HTML pages (such as background pages)

Google chrome extensions

Development: manifest.json

- Extension metadata - defines the extension, its attributes, permissions, page/browser actions, additional files, etc

```
   {
// Required
  "name": "My First Extension",
  "version": "1.0",

  // Optional (many more attributes than shown)
      "icons": {"128":"icon128.png"},
  "homepage_url": "http://cs.cf.ac.uk/",
  "description": "An example extension."
   }
```

- ... great, but this doesn't really *do* anything...

Google chrome extensions

Development: Actions

... we need actions! These enable extensions to do stuff

- **Browser actions** (front-ended with icon & popup)
- Page actions (icons in address bar e.g. RSS)
- Desktop notifications (e.g. new email)
- Options pages (to change the extension behaviour)
- **Override pages** (override history & new tabs page)
- **Content Scripts** (inject CSS or JavaScript into a page)

Development: Permissions

- Tell Chrome what permissions you need
- Defined in manifest:

```
"permissions": ["tabs", "http://api.flickr.com/"]
```

- Used to
  - Access APIs (Facebook, Twitter, etc.)
  - Access Chrome properties (tabs, history, etc.)
  - Provide notifications

Development: Browser Action

- Add a "browser_action" attribute to the manifest...

```
"browser_action": {
   "default_icon": "icon.png",
   "default_title": "Click to show popup (tooltip)",
   "popup": "popup.html"
}
```

- ... and some icons

Development: Content Scripts

- Add a "content_scripts" attribute to the manifest...

```
"content_scripts": [ {
     "css": ["styles.css"],
     "js": ["script.js"],
     "matches": ["http://*.twitter.com/*",
                 "http://twitter.com/*"],
     "run_at": "document_end"
} ]
```

- Applies new styles or scripts to the page

- Only runs the scripts at pre-specified pages

Development: Chrome API

- Series of methods exposed to all Chrome extensions
- Can be run in any page involved in the extension
- Fully JavaScript-based
- Need to give the manifest permissions for each "package" needed:

```
"permissions": ["history"]
```

Development: Chrome API

- Examples of Chrome APIs:
  - chrome.**bookmarks**
  - chrome.**tabs**
  - chrome.**windows**
  - chrome.**omnibox**
  - chrome.**cookies**
  - chrome.**extension**

- 13 supported APIs in total. Each has a set of methods and events

- For each used, add to "permissions" in manifest

Google chrome extensions

Development: Chrome API

- For example, chrome.**browserAction.**
    - setIcon
    - setPopup
    - setTitle

- Can also use Event methods:

chrome.**browserAction**.onClicked.addListener(function);

Google chrome extensions

Development: Override page

- Add a "chrome_url_overrides" attribute to the manifest...

```
"chrome_url_overrides": {
   "overriding_page": "new_page.html"
}
```

- Overridable pages
  - newtab
  - history
  - bookmarks

Development: Background Pages

- Run as a single, long-living script
- Alive for the lifetime of the extension
- Can be used to provide notifications
  - Need "background" permission in manifest
- Usually used in more complex extensions

Development: Themes

- Basically, a type of extension. Example manifest:

```
{
 "version": "2.6",
 "name": "Example Theme",
 "theme": {
   "images" : {
     "theme_frame" : "images/theme_frame.png",
     "theme_frame_overlay" : "images/theme_frame_stripe.png", etc...
   },
   "colors" : {
     "frame" : [71, 105, 91],
     "toolbar" : [207, 221, 192],
     "button_background" : [255, 255, 255]
   },
   "tints" : {
     "buttons" : [0.33, 0.5, 0.47]
   }
 }
}
```

Development: Apps

- Again, a form of extension
- Only really contain a link to the web app

```
"app": {
    "launch": {
        "web_url": "http://www.my_app.com/"
    },
    "icons": {set_of_icons},
    "permissions": [array_of_permissions]
}
```

Summary

- Manifest file
- Browser actions
- Content scripts
- Override pages
- Chrome API
- Background pages
- Themes & apps

Google chrome extensions

Thank you for listening

code.google.com/chrome/extensions/