mozilla

Extension Bootcamp

Zero to “Hello world!” In 45(*2) Minutes

Myk Melez — myk@mozilla.org
Design Challenge — 2009 March 13
http://people.mozilla.com/~myk/bootcamp/

Your Mission, Should You Choose to Accept It

- learn what extensions are, what they can do, the stuff they're made of
- configure your development environment
- make an extension
- package and test it

Required Gear

- a Linux, Mac, or Windows machine
- a terminal application (GNOME Terminal, Terminal, Command Prompt)
- a text editor (Text Editor, TextEdit, Notepad, or Komodo Edit on all three platforms)
- a ZIP archiver (zip, 7-Zip)
- basic knowledge of HTML, CSS, JavaScript

Rules of Engagement

- this is not a military bootcamp
- I won't be barking orders
- you don't have to do push-ups
- questions are encouraged!
  - press the Fast button in WebEx if I'm talking too fast
  - ask questions in Q&A (my colleagues will monitor and respond)
  - contact me afterwards in the group or on IRC to go over something again or for more details

Step One: Shave Your Head

http://www.flickr.com/photos/soldiersmediacenter/2218951540/

What's an extension?

- a package of files that modifies Firefox's appearance and/or behavior
- in the ZIP archive format (but with a .xpi file extension)
- two manifest files describe contents
- distribute, browse, search via addons.mozilla.org
- install, manage, uninstall via Add-ons window

What can they do?

- add and remove interface elements (menus, buttons, etc.)
- modify appearance of elements (color, border, icons, etc.)
- listen and respond to events (page loads, mouse clicks, etc.)
- access modules/components (file manipulation, networking, data storage, etc.)
- add their own modules/components and override the built-in ones

Content vs. Chrome

- **content** is web pages that Firefox loads in browser tabs
- **chrome** is Firefox's user interface (i.e. everything around those browser tabs, plus other windows like Preferences and Add-ons)
- **chrome privileges** are the ability to do anything that Firefox can do
- extensions are part of chrome and have chrome privileges!

Phenomenal Cosmic Power!

http://adolwyn.livejournal.com/

Itty Bitty Living Space

http://adolwyn.livejournal.com/

Extension Building Blocks

- the right tool for the job!
- **XUL** and **XHTML** for structure, widgets
- **CSS** for appearance, style
- **JavaScript** for behavior
- **DTDs** and **properties files** for localization
- **JavaScript** and **C++** for modules/components

XUL

- **X**ML **U**ser-interface **L**anguage
- XML vocabulary for building interfaces
- like HTML, but for applications
- tags like <menu>, <button>, and <tab>
- different layout (box) model
- much of Firefox's interface is built with XUL
- you can mix XUL and XHTML

Development Environment

- Firefox profile
- developer preferences
- developer extensions
- extension directory

Profiles

- a hidden feature of Firefox
- directories that store user data (preferences, bookmarks, saved passwords, etc.)
- extensions are profile-specific
- Firefox uses one profile by default
- a separate profile for development isolates changes and protects data in your regular profile
- create a profile by creating a directory:
  - mkdir -p ~/Profiles/helloworld-dev
  - md "%USERPROFILE%\Profiles\helloworld-dev"

Using a Profile

- run Firefox with the **-profile** command line argument
  - /usr/bin/firefox -profile ~/Profiles/helloworld-dev
  - /Applications/Firefox.app/Contents/MacOS/firefox -profile ~/Profiles/helloworld-dev
  - "C:\Program Files\Mozilla Firefox\firefox.exe" -profile "%USERPROFILE%\Profiles\helloworld-dev"
- use two profiles at the same time
  - **-no-remote** command line argument
  - **MOZ_NO_REMOTE=1** environment variable

Developer Preferences

- make it easier to develop extensions
- editable via **about:config**
- **extensions.logging.enabled**: shows extension install/update errors in Error Console
- **javascript.options.showInConsole**: shows extension code errors in Error Console
- **browser.dom.window.dump.enabled**: lets extensions use the **dump** function in JavaScript to print messages to standard console
- set them all to **true**!

Developer Extensions

- **DOM Inspector** lets you examine the structure and state of Firefox's chrome:

  https://addons.mozilla.org/en-US/firefox/addon/6622

- **Console²** separates chrome and content errors to make it easier to catch problems in your code:

  https://addons.mozilla.org/en-US/firefox/addon/1815

Extension Directory

- the place where you put extension files
- you can call it anything you want
- you can put it anywhere you want
- I put mine in ~/Projects/:
  - mkdir -p ~/Projects/helloworld
  - md "%USERPROFILE%\Projects\helloworld"

Basic File Layout

- helloworld/
  - install.rdf
  - chrome.manifest
  - chrome/
  - content/
  - locale/
    - en-US/

The Install Manifest

- tells Firefox about the extension (its name, with which versions of Firefox it is compatible, etc.)
- written in RDF, an XML vocabulary
- must be at top level of extension directory
- must be called **install.rdf**

A “Simple” Install Manifest

```
<?xml version="1.0" encoding="UTF-8"?>
<RDF xmlns="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:em="http://www.mozilla.org/2004/em-rdf#">
  <Description about="urn:mozilla:install-manifest">
    <em:id>helloworld@myk.melez</em:id>
    <em:name>Hello World!</em:name>
    <em:version>0.1</em:version>
    <em:targetApplication> <!-- Firefox -->
      <Description>
        <em:id>{ec8030f7-c20a-464f-9b0e-13a3a9e97384}</em:id>
        <em:minVersion>3.0</em:minVersion>
        <em:maxVersion>3.2a1pre</em:maxVersion>
      </Description>
    </em:targetApplication>
  </Description>
</RDF>
```

http://people.mozilla.com/~myk/bootcamp/install.rdf

The Chrome Manifest

- tells Firefox about the contents of the extension (the kinds of files it contains and where they are located)
- written in a simple line/space-delimited format
- must be at top level of extension directory
- must be called **chrome.manifest**

A Simple Chrome Manifest

```
content  helloworld         chrome/content/
locale   helloworld  en-US  chrome/locale/en-US/

overlay  chrome://browser/content/browser.xul chrome://helloworld/content/browser.xul
```

http://people.mozilla.com/~myk/bootcamp/chrome.manifest

XUL Overlays

- a way to insert your XUL into Firefox's XUL
- add elements, style, script, localization strings
- chrome.manifest instructions tell Firefox which overlay to insert into which Firefox XUL file

chrome: URLs

- reference files inside Firefox application and extensions
- require instruction in chrome.manifest file
- Firefox translates them to locations on disk
- chrome://helloworld/content/browser.xul == helloworld/chrome/content/browser.xul

A Simple XUL Overlay

```
<overlay xmlns="http://www.mozilla.org/keymaster/gatekeeper/there.is.only.xul">
  <button label="Hello world!"/>
</overlay>
```

http://people.mozilla.com/~myk/bootcamp/browser.xul

Installing Your Extension: Link File Method

- create a text file inside the **extensions/** subdirectory of your development profile
- the name of the file must be the ID of your extension (f.e. **helloworld@myk.melez**)
- the contents of the file must be a single line specifying the path to your extension directory:
  - /home/myk/Projects/helloworld/
  - /Users/myk/Projects/helloworld/
  - C:\Documents and Settings\myk\Projects\helloworld\
  - C:\Users\myk\Projects\helloworld\

Testing Your Extension

- start Firefox
- it should open the Add-ons window and tell you that your extension was installed
- a **Hello world!** button should appear at the bottom of the browser window

Troubleshooting

- the Add-ons window doesn't open
  - the link file's name might not match the ID of your extension
  - the path inside the link file might not point to your extension directory
    - install.rdf might not be readable or contain necessary information
    - chrome.manifest might not be registering the overlay
- an XML error window opens
  - the <button> might not contain a closing slash

Are we having fun yet?

http://flickr.com/photos/soldiersmediacenter/3976148555/

Putting It Somewhere Specific

- surround it with a tag representing its parent
- use **insertbefore** or **insertafter** to specify its position relative to its siblings

```
<overlay xmlns="http://www.mozilla.org/keymaster/gatekeeper/there.is.only.xul">
  <statusbar id="status-bar">
    <button label="Hello world!" insertafter="statusbar-display"/>
  </statusbar>
</overlay>
```

http://people.mozilla.com/~myk/bootcamp/browser-2.xul

Testing Your Changes

- restart Firefox
- the button should be inside the statusbar after the part of the statusbar that displays messages

Troubleshooting

- the button has disappeared
  - did you give the **<statusbar>** tag the right ID?

Making It Do Something

- event handler attributes
- inside a XUL **<script>** tag
- external file referenced by a XUL **<script>** tag

```
<overlay xmlns="http://www.mozilla.org/keymaster/gatekeeper/there.is.only.xul">
  <script type="application/javascript"
          src="chrome://helloworld/content/browser.js"/>
  <statusbar id="status-bar">
    <button label="Hello world!" insertafter="statusbar-display"
            oncommand="HelloWorld.onCommand(event)"/>
  </statusbar>
</overlay>
```

http://people.mozilla.com/~myk/bootcamp/browser-3.xul

External Script

- put into **chrome/content/**
- define functions and variables inside an object to minimize the risk of conflicting with those defined by other extensions or core Firefox code

```
let HelloWorld = {
  onCommand: function(event) {
    dump("Hello world!\n");
  }
};
```

http://people.mozilla.com/~myk/bootcamp/browser.js

Testing Your Changes

- restart Firefox from the terminal (Windows users: add **-console** argument to command)
- press the button
- the text “Hello world!” should appear in the console

Troubleshooting

- “Hello world!” doesn't appear in the console
  - did you set **browser.dom.window.dump.enabled** to true?
  - on Windows, did you start Firefox with the **-console** argument?
  - does **Tools > Error Console** report a JavaScript error?

Giving It Style

- interface elements in Firefox are styled using CSS
- mostly it works the same as with HTML web pages
- there are also some Mozilla-specific selectors and properties
- sometimes the OS's style overrides CSS (workaround: set **-moz-appearance: none** in CSS)

Including CSS

- the **style** attribute to XUL/XHTML tags
- external file referenced by an **xml-stylesheet** processing instruction

```
<?xml-stylesheet href="chrome://helloworld/content/browser.css" type="text/css"?>
<overlay xmlns="http://www.mozilla.org/keymaster/gatekeeper/there.is.only.xul">
  <script type="application/javascript"
          src="chrome://helloworld/content/browser.js"/>
  <statusbar id="status-bar">
    <button id="helloWorldButton" label="Hello world!"
            insertafter="statusbar-display" oncommand="HelloWorld.onCommand(event)"
            style="color: green"/>
  </statusbar>
</overlay>
```

http://people.mozilla.com/~myk/bootcamp/browser-4.xul

External Stylesheet

- a regular CSS stylesheet
- put into **chrome/content/**

```
#helloWorldButton {
   font-weight: bold;
}
```

http://people.mozilla.com/~myk/bootcamp/browser.css

Testing Your Changes

- restart Firefox from the terminal
- the text of the button should be green and bold

Troubleshooting

- the text of the button isn't green
  - is the CSS rule in the **style** attribute correct?
- the text of the button isn't bold
  - did you put the stylesheet into **chrome/content/**?
  - did you specify the stylesheet via the **xml-stylesheet** processing instruction?
  - did you set the button's **id** attribute to **helloWorldButton**?

Almost There!

http://www.flickr.com/photos/soldiersmediacenter/3976168501/

Going Local

- combination of technologies for localizing interface
- DTDs provide strings that appear in the interface from the moment it is loaded and don't change
- properties files provide strings that can be inserted into interface (or removed from it) dynamically while Firefox is running
- each language has its own set of locale files
- Firefox picks the right set automatically based on each user's locale

External DTD

- name/value pairs
- uses special **!ENTITY** tags
- put into **chrome/locale/en-US/**

```
<!ENTITY  helloWorldButton.label  "Hello world!">
```

http://people.mozilla.com/~myk/bootcamp/browser.dtd

Including a DTD

- include external file via DOCTYPE declaration
- insert string from DTD via character entity

```
<?xml-stylesheet href="chrome://helloworld/content/browser.css" type="text/css"?>
<!DOCTYPE overlay SYSTEM "chrome://helloworld/locale/browser.dtd">
<overlay xmlns="http://www.mozilla.org/keymaster/gatekeeper/there.is.only.xul">
  <script type="application/javascript"
          src="chrome://helloworld/content/browser.js"/>
  <statusbar id="status-bar">
    <button id="helloWorldButton" label="&helloWorldButton.label;"
            insertafter="statusbar-display" oncommand="HelloWorld.onCommand(event)"
            style="color: green"/>
  </statusbar>
</overlay>
```

http://people.mozilla.com/~myk/bootcamp/browser-5.xul

Test Your Changes

- restart Firefox
- the button should still say “Hello world!”

Troubleshooting

- Firefox displays an undefined entity error
  - did you put the DTD file in **chrome/locale/en-US/**?
  - did you include the DTD via a DOCTYPE declaration?
  - did you define the en-US localization via chrome.manifest?
  - are you using the en-US version of Firefox?
- the button says “helloWorldButton.label”
  - did you prefix the name of the entity with an ampersand (&) and suffix it with a semi-colon (;)?
- the button has disappeared
  - are you referencing the DTD at the right location?

External Properties File

- name/value pairs
- values can contain placeholders that will be replaced by other strings (**%1$S**, **%2$S**, etc.)
- put into **chrome/locale/en-US/**

```
helloWorld = %1$S world!
morningGreeting = Good morning
eveningGreeting = Good evening
```

http://people.mozilla.com/~myk/bootcamp/browser.properties

Including a Properties File

- include external file via **<stringbundle>** tag

```
<?xml-stylesheet href="chrome://helloworld/content/browser.css" type="text/css"?>
<!DOCTYPE overlay SYSTEM "chrome://helloworld/locale/helloworld.dtd">
<overlay xmlns="http://www.mozilla.org/keymaster/gatekeeper/there.is.only.xul">
  <stringbundleset id="stringbundleset">
    <stringbundle id="helloWorldBundle"
                  src="chrome://helloworld/locale/browser.properties"/>
  </stringbundleset>
  <script type="application/javascript"
          src="chrome://helloworld/content/browser.js"/>
  <statusbar id="status-bar">
    <button label="&helloWorldButton.label;" insertafter="statusbar-display"
            oncommand="HelloWorld.onCommand(event)"
            style="color: green"/>
  </statusbar>
</overlay>
```

http://people.mozilla.com/~myk/bootcamp/browser-6.xul

Using a Properties File

- access properties file via string bundle API
- get string bundle via DOM methods
- use **getString** to get a simple string from bundle
- use **getFormattedString** to get a string with placeholders from bundle

```
let HelloWorld = {
  onCommand: function(event) {
    let stringBundle = document.getElementById("helloWorldBundle");
    let hours = new Date().getHours();
    let greeting = (hours < 12) ? stringBundle.getString("morningGreeting")
                                : stringBundle.getString("eveningGreeting");
    let message = stringBundle.getFormattedString("helloWorld", [greeting]);
    dump(message + "\n");
  }
};
```

http://people.mozilla.com/~myk/bootcamp/browser-2.js

Test Your Changes

- restart Firefox
- press button
- the text “Good morning world!” or “Good evening world!” should appear in the console

Troubleshooting

- the text doesn't appear in the console
  - did you set **browser.dom.window.dump.enabled** to true?
  - on Windows, did you start Firefox with the **-console** argument?
  - does **Tools > Error Console** report a JavaScript error?

Packaging Your Extension

- packaging is easy!
- packages are ZIP files (but with **.xpi** extensions)
- you need a ZIP archiver
- change to the extension directory
- create a ZIP archive of the files you've created
  - Linux/Mac: zip -r helloworld.xpi install.rdf chrome.manifest chrome
  - Windows: 7za a -tzip -r helloworld.xpi install.rdf chrome.manifest chrome

Testing Your Extension

- create another profile
  - mkdir -p ~/Profiles/helloworld-test
  - md "%USERPROFILE%\Profiles\helloworld-test"
- start Firefox with that profile
  - /usr/bin/firefox -profile ~/Profiles/helloworld-test
  - /Applications/Firefox.app/Contents/MacOS/firefox -profile ~/Profiles/helloworld-test
  - "C:\Program Files\Mozilla Firefox\firefox.exe" -profile "%USERPROFILE%\Profiles\helloworld-test"
- open the package you created: **File > Open File...**
- press the **Install Now** button
- press the **Restart Firefox** button

Troubleshooting

- Firefox reports an error installing your extension
  - did you archive the extension directory itself instead of its contents?

Congratulations! You are an extension developer.

http://www.flickr.com/photos/soldiersmediacenter/2541459436/

Further Topics

- the XUL box model
- using built-in and third party JavaScript modules and XPCOM components
- writing your own modules and components
- making your extension's appearance OS-specific
- additional developer preferences (f.e. nglayout.debug.disable_xul_cache)
- additional developer extensions (f.e. Extension Developer's Extension, Chromebug)
- unit testing frameworks

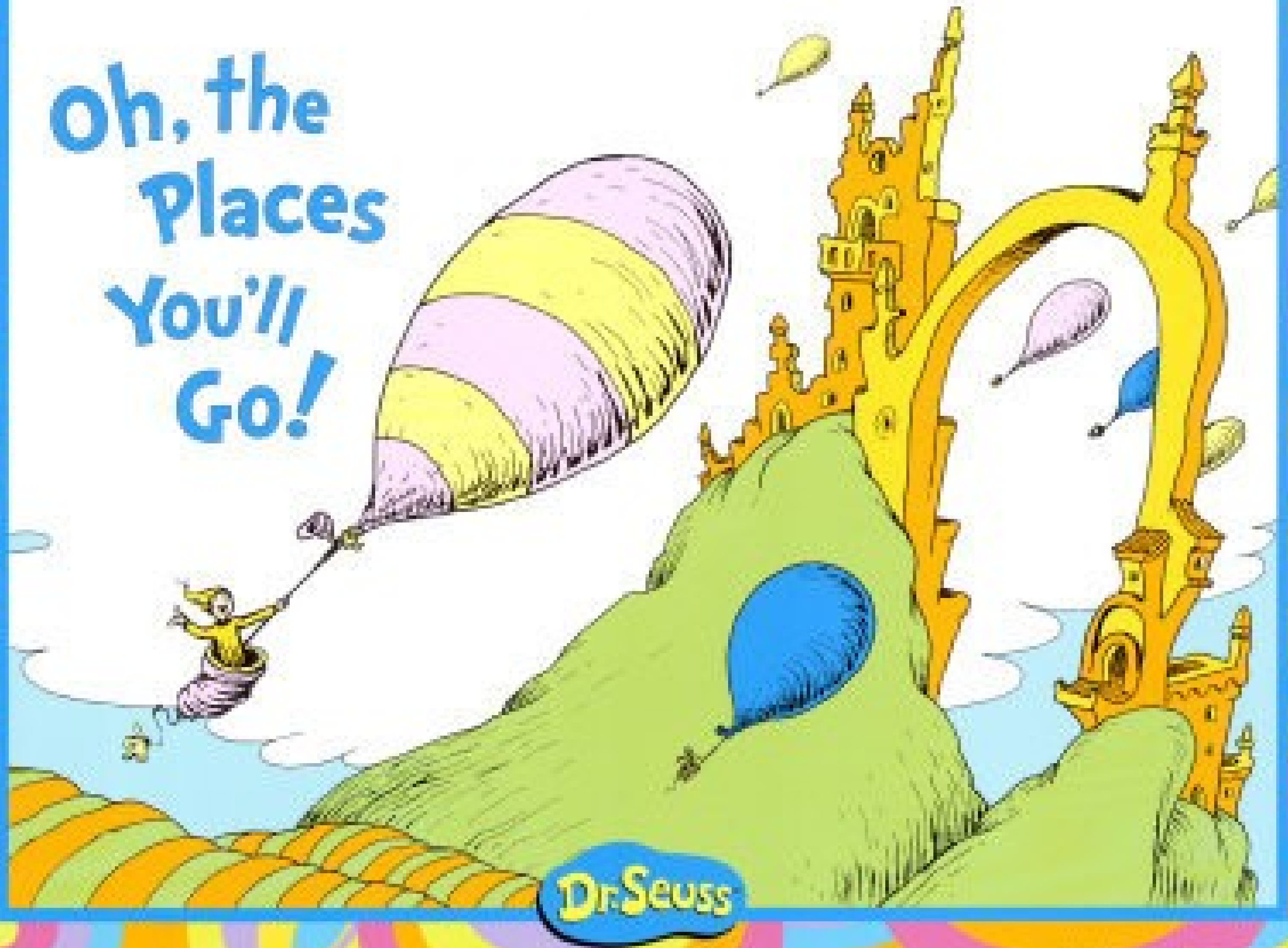
Getting Help

- Mozilla Developer Center
  http://developer.mozilla.org/

- mozilla.dev.extensions discussion group
  <http://groups.google.com/group/mozilla.dev.extensions>

- #extdev and #labs IRC channels on irc.mozilla.org
  <ircs://irc.mozilla.org:6697/extdev>
  <ircs://irc.mozilla.org:6697/labs>

Oh, the Places You'll Go!



Dr. Seuss