LEARNING firefox-addon

#firefox-addon

Table of Contents

About

Chapter 1: Getting started with firefox-addon

Examples

Introduction

Add-ons:

Firefox add-ons are generally grouped into Extensions, and then "other types" of Firefox add-ons.

Extensions

Extensions allow Firefox to be customized by adding to or modifying the functionality of Firefox. Some of the types of things which can be done with extensions include:

- Change how specific websites appear, their content, or how they are interacted with.
- Customize the Firefox user interface
- Add additional features to Firefox
- Change how existing Firefox features function

Firefox extensions are, primarily, written in JavaScript with the addition of some JavaScript APIs.

Deprecation and removal of all types of extensions other than WebExtensions

Firefox add-ons, particularly extensions, are in a state of flux at the moment. Mozilla has announced, and confirmed, that they have deprecated all types of Firefox extensions, except WebExtensions, and that all non-WebExtensions based extensions will be disabled in Firefox 57, which is scheduled for 2017-11-14.

Types of Extensions

Firefox has four types of extensions (all of which are commonly referred to as add-ons):

- WebExtensions: Moving forward, WebExtensions are the only type of Firefox extension which will be supported. These add-ons are described by a *manifest.json* file. This API is similar to what is used for Google Chrome extensions. These add-ons use HTML and CSS in addition to Javascript. While Mozilla has stated that this API is the future of Firefox extensions, this API is still in development. For now, you are probably best off developing and testing your WebExtension add-on with Firefox Developer Edition, or Firefox Nightly. You should also make careful note of what version of Firefox is required for the functionality you desire to use. This information is contained in the "Browser compatibility" section of the MDN documentation pages.

WebExtensions use a significantly different API than the other three types of extensions. There is, intentionally, no ability to use the interfaces provided by any of the other add-on types.

- Add-on SDK: [**deprecated; scheduled for removal**] These add-ons are described by a *package.json* file which is initially generated by executing `jpm init`. These extensions will often use `require()` to load either High-Level, or Low-Level APIs to interface with Firefox. These add-ons use HTML and CSS in addition to Javascript. Currently, these add-ons are wrapped into a bootstrapped extension when they are loaded for testing by `jpm run` or consolidated into an *.xpi* file by `jpm xpi` for distribution (i.e. upload to AMO/Mozilla). In other words, they are bootstrapped extensions with an SDK wrapper.

  Mozilla appears to be committed to continuing to support Add-on SDK based extensions as long as the extension does not use `require("chrome")`, or otherwise depend on XUL, XPCOM , and XBL.

  Most of the things that can be done in a bootstrapped extension can be done in an Add-on SDK based one. However, many such things bypass the SDK which forfeits a significant portion of the benefits of using the Add-on SDK.

- Bootstrapped: [**deprecated; scheduled for removal**] These extensions are also commonly called "restartless" because they were the first type of Mozilla extension which did not require the application to be restarted in order to load/unload the add-on. However, restartless is a descriptor of how they function. Using "restartless" as the name for this type of add-on is confusing because both Add-on SDK and WebExtension add-ons also do not require the application to be restarted upon load or unload of the add-on. For that reason, there is a tendency to no longer use "restartless" as the name for this type of add-on.

  These add-ons use HTML and CSS in addition to Javascript. Many also interact with Firefox using XUL.

  These add-ons have a JavaScript file called *bootstrap.js* which must contain entry points (functions) which are called for add-on `startup()`, `shutdown()`, `install()` and `uninstall()`.

  These add-ons contain a *install.rdf* file that describes the add-on. They usually, but not always, also contain a *chrome.manifest* file that describes how the files and directories in the extension relate to the Mozilla application (e.g. Firefox).

  Most, but not all, of the things that can be done in overlay/XUL/Legacy extensions can be accomplished in bootstrapped add-ons. Anything that can be done in the Add-on SDK can be done in a bootstrapped extension (Add-on SDK extensions are bootstrapped add-ons with some JavaScript based API layers).

  Mozilla has stated that they plan to deprecate "add-ons that depend on XUL, XPCOM, and XBL." While not all bootstrapped add-ons depend on these technologies, there is a tendency for for bootstrapped add-ons to operate at a lower level than Add-on SDK and WebExtension add-ons. Thus, they are more likely to use these technologies. While there are some that are saying that all bootstrapped add-ons are planned to be deprecated, it is not clear that is the case. After all, Add-on SDK extensions are not being deprecated (unless they use `require("chrome")`, or otherwise depend on XUL, XPCOM, or XBL) and all Add-on SDK extensions are bootstrapped extensions, just with an SDK wrapper.

- Overlay/XUL/Legacy: [**deprecated; scheduled for removal**] These add-ons contain a *install.rdf* file that describes the add-on and a *chrome.manifest* file to describe how the add-on's files relate to (e.g. overlay) the application's files. How the add-on functions with the application is completely dependent on the relationships described in the *chrome.manifest* file. The only exceptions to this are a few things like icons for the extension and the file describing the extension's options which are indicated in the *install.rdf* file. These extensions interact with the application (e.g. Firefox) at a very low level. This tends to make them more likely to break when changes are made to the application.

  These add-ons use XUL, HTML and CSS in addition to Javascript. Some also use XPCOM, and XBL.

  All Overlay/XUL/Legacy extensions are planned to be deprecated.

Other Types of Firefox Add-ons

When most people think about Firefox add-ons they are thinking about the extensions described above. However, there are some additional types of Firefox add-ons:

- Themes [**partially deprecated**] allow for the customization of the look and feel of Firefox. Primarily by providing different CSS rules to all parts of the browser. "Complete themes" are deprecated and planed to be partially replaced with a currently non-existent interface.
- Mobile add-ons are for Firefox for Android. Currently, all Firefox extension types which are restartless are supported.
- Search engine plugins are used to add additional search engines to the browser's search bar.
- User dictionaries allow using spell-check in additional languages.
- Language packs add additional languages to the Firefox user interface.
- Plugins are shared libraries to display content that the application itself can't display natively.

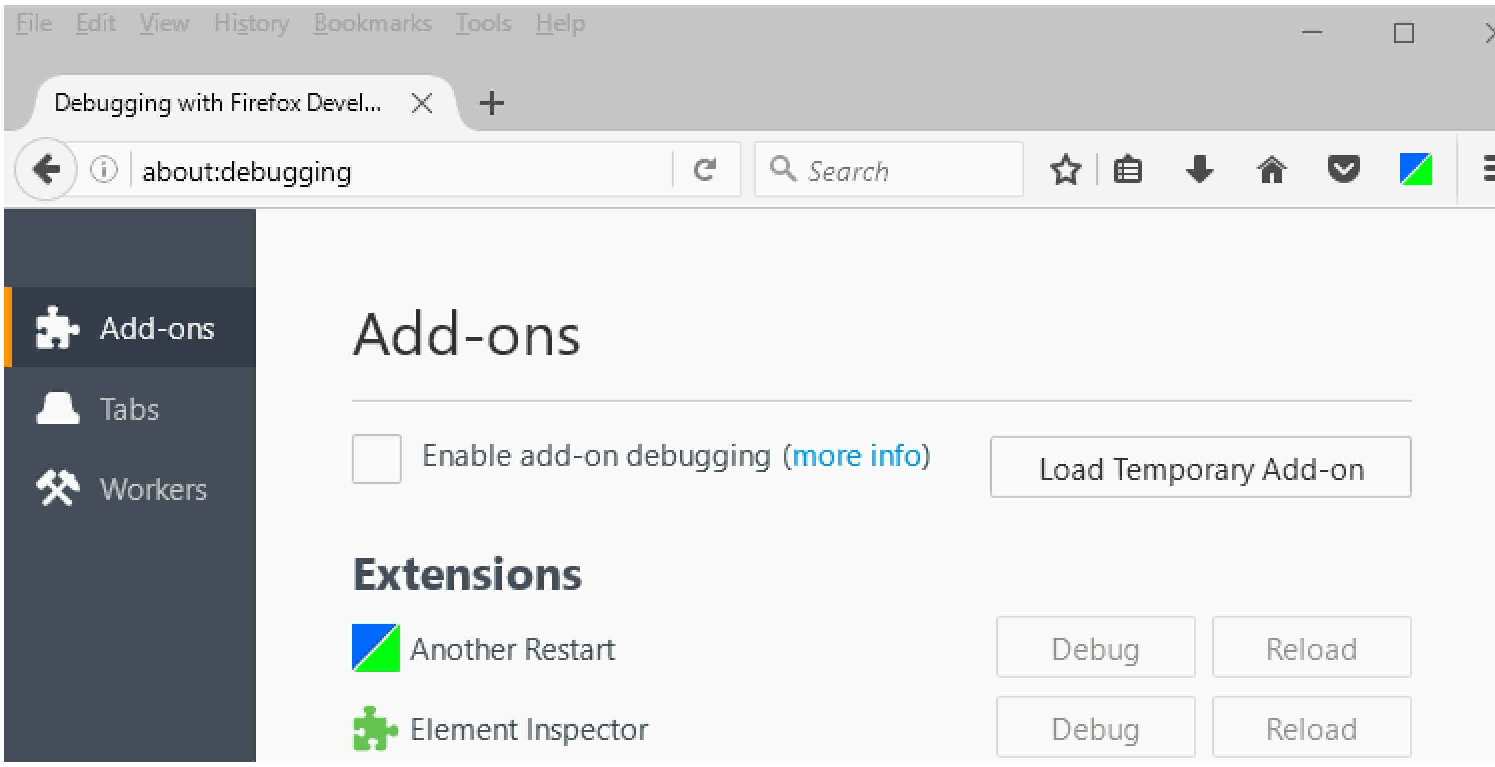
This "example" is primarily copied, with some modifications, from my, Makyen's, answer on a stackoverflow question. Some portions of this content were copied, or at least based upon, the Add-on page on Mozilla Developer Network (MDN).
This was originally posted by Makyen in the "Remarks" section of the firefox-addon tag. It was then modified by Ondřej Doněk, who removed an extra "for ". In a single edit performed by nus, it was moved from "Remarks" to a pinned "example" titled "Introduction". Unfortunately, doing so resulted in the system loosing attribution information.

Installing a Temporary Add-on

To test an add-on you are developing, you will likely desire to install it in Firefox temporarily. You can do so by loading it as a Temporary Add-on. To do so:

1. Go to `about:debugging`
2. Click on "Load Temporary Add-on"
3. In the file picker, navigate to the directory containing the add-on files
4. Select any file in the folder
5. Click "Open"

The following animation shows loading an add-on named "aaaaaaaaaaaaaaaaaa - demo add-on" from `about:debugging` and that the add-on then shows up in `about:addons`:


File Edit View History Bookmarks Tools Help
Debugging with Firefox Devel...
about:debugging
Search
Add-ons
Tabs
Workers

Add-ons

Enable add-on debugging (more info)

Load Temporary Add-on

Extensions

Another Restart Debug Reload

Element Inspector Debug Reload

As a Temporary Add-on, you can load either an unpacked add-on (a directory which contains all files for the add-on which you would pack into an *.xpi* file), or an add-on which is packed into an `.xpi` archive. Temporary Add-ons do not need to be signed. Temporary Add-on remain installed until manually uninstalled, or Firefox restarts.

Mozilla documentation: Temporary Installation in Firefox

WebExtensions

WebExtensions can be loaded as Temporary Add-ons. This can be done with the add-on files either unpacked, or packaged in an *.xpi* file.

Firefox Add-on SDK

You can not load a Firefox Add-on SDK extension as a temporary add-on without first packaging it into an *.xpi* file with `jpm xpi`. In general, you will use `jpm run` to test your Firefox Add-on SDK extension.

The files which are typically edited for an Add-on SDK extension do not make a complete extension without some additional wrapping functions and the *package.json* file being translated into an *install.rdf* file and, possibly, a *chrome.manifest* file. This process wraps the Add-on SDK extension into a Bootstrap/Restartless add-on, which is understood by Firefox. Without this process, Firefox will not be able to load the add-on. This process is performed by `jpm xpi` resulting in a packed *.xpi* file. Executing `jpm run` also performs this process, but stores the resulting files in a temporary location and invokes Firefox with the add-on installed.

Bootstrap/Restartless

Bootstrap/Restartless add-ons can be loaded as Temporary add-ons. This can be done with the add-on files either unpacked, or packaged in an *.xpi* file.

Legacy/Overlay/XUL

Legacy/Overlay/XUL add-ons can **not** be loaded as temporary add-ons.

Installing unsigned add-ons

In order to install an extension as a normal add-on into Release or Beta versions of Firefox greater than, or equal to, version 48, the extension *must* be signed by Mozilla. An extension is signed by submitting it to AMO. Once it is signed, the extension can be installed on any version of Firefox which it supports. For Firefox versions prior to version 43, extensions were not required to be signed by Mozilla. Types of add-ons other than extensions are not required to be signed by Mozilla.

You can install unsigned extensions as normal add-ons into other versions of Firefox (e.g. Firefox Developer Edition, Firefox Nightly, Unbranded Beta, or Unbranded Release) by setting `xpinstall.signatures.required` to `false` in `about:config`. Setting this option was also effective in Release and Beta Firefox versions 43–48. Setting this option is not effective in Release and Beta versions of Firefox from version 48 onward.

The need to be able to install unsigned add-ons during add-on development has been greatly reduced by the availability of installing extensions as Temporary Add-ons. Temporary Add-ons do not need to be signed, and can be loaded into any current version of Firefox. As the name implies, the primary drawback of Temporary Add-ons is that they are temporary. They must be re-installed any time Firefox is restarted. However, there are use cases where it is preferable to install an add-on as a normal, but unsigned, add-on instead of as a Temporary Add-on. An add-on should be installed as an unsigned add-on if there is need for the add-on to remain installed even after Firefox is restarted. This could be desirable for a variety of reasons, including: longer term use testing, or to test how the add-on functions when Firefox starts up.

Installing add-ons for development

Add-ons can be installed as:

1. Normal add-ons, which are installed until uninstalled

2. Temporary Add-ons (extensions only): are only installed until Firefox is restarted, or can manually uninstalled earlier.
3. Using `jpm run` (Add-on SDK only): Automatically runs Firefox using a temporary profile with your add-on loaded.
4. Using `web-ext run` (WebExtensions only): Automatically runs Firefox using a temporary profile with your add-on loaded as a temporary add-on. By default, monitors your extension files for changes and automatically reloads your extension when files change.

Normal Add-ons

Installing packaged extensions (i.e. the `.xpi` file) can be a simple matter of dragging and dropping it onto a Firefox window running the profile in which you desire it installed. They can also be installed directly by downloading the extension from AMO. Depending on what your goal is (one profile, all profiles, all users, which OS, etc.), there are other options as to how to install extensions .

These other options include various directories outside the profile's directory into which you can place the *.xpi* file to have it be installed for all users of a particular version of Firefox, or all profiles of a particular user. On Windows, you can also install an extension by adding a key to the Windows Registry. In general, these other directories are not ones into which you would install an add-on on which you are currently writing. However, they can be used to make sure you have add-ons you use to support your testing/development loaded in any Firefox profile or Firefox version which you use. For instance, by placing an *.xpi* file in `<Firefox install directory>/browser/extensions` you can have an extension available even in the temporary profile created by `jpm run` (used for testing Firefox Add-on SDK based extensions).

For development/testing, you can have the extension be in any directory on your local drive by using a Firefox extension proxy file (create a file named as the extension's `<em:id>` (in *install.rdf* for Bootstrap/Restartless and Overlay/Legacy) in the profile's *extensions* directory containing one line with the complete path to the directory containing the extension's files). Extensions installed in this way will almost always be unsigned (see below). Thus, this method is not very useful if you want to install the extension into a Release or Beta version of Firefox.

Limitations on installing normal add-ons: Add-on Signing

In order to install an extension as a normal add-on into Release or Beta versions of Firefox greater than, or equal to, version 48, the extension *must* be signed by Mozilla. An extension is signed by submitting it to AMO. Once it is signed, the extension can be installed on any version of Firefox which it supports. For Firefox versions prior to version 43, extensions were not required to be signed by Mozilla. Types of add-ons other than extensions are not required to be signed by Mozilla.

You can install unsigned extensions as normal add-ons into other versions of Firefox (e.g. Firefox Developer Edition, Firefox Nightly, Unbranded Beta, or Unbranded Release) by setting `xpinstall.signatures.required` to `false` in `about:config`. Setting this option was also effective in Release and Beta Firefox versions 43–48. Setting this option is not effective in Release and Beta versions of Firefox from version 48 onward.

However, you can completely disable add-on signature checking in Firefox on all versions, including release. The answer to the Stack Overflow question How can I disable signature checking for Firefox add-ons? describes how to do so.

The need to be able to install unsigned add-ons for development purposes has been greatly reduced by the availability of installing extensions as Temporary Add-ons. Temporary Add-ons do not need to be signed, and can be loaded into any current version of Firefox. As the name implies, the primary drawback of Temporary Add-ons is that they are temporary. They must be re-installed any time Firefox is restarted. However, there are use cases where it is preferable to install an add-on as a normal, but unsigned, add-on instead of as a Temporary Add-on. An add-on should be installed as an unsigned add-on if there is need for the add-on to remain installed even after Firefox is restarted. This could be desirable for a variety of reasons, including: longer term use testing, or to test how the add-on functions when Firefox starts up.

Temporary Add-ons (extensions only)

WebExtensions, Firefox Add-on SDK based extensions, and Restartless/Bootstrap extensions can be installed as Temporary Add-ons from `about:debugging`. Extensions can be loaded as Temporary Add-ons in any current version of Firefox. Temporary Add-ons are not required to be signed and can be loaded from either unpacked (a directory with files), or packed (e.g. a *.xpi* file).

For detailed information on temporarily installing extensions, see Installing a Temporary add-on.

Read Getting started with firefox-addon online: https://riptutorial.com/firefox-addon/topic/3235/getting-started-with-firefox-addon

Credits

S. No	Chapters	Contributors
1	Getting started with firefox-addon	Community, Makyen, Martin Zhai, nus, Ondřej Doněk, Priya