mozilla LABS

Jetpack

Using the Add-on SDK
v.gd/usingaddonsdk

Genesis

- it's too hard to make and use add-ons...
- most potential developers don't know XUL
- Firefox updates can break add-ons
- installing add-ons requires restarting Firefox
- add-ons have full control over system, even though most don't need it

mozilla LABS
Jetpack

Goals

- simplify add-on development via high-level APIs that use common web technologies
- keep add-ons working across Firefox upgrades (Write Once, Run Everywhen!)
- make add-ons installable/removable without a restart
- limit add-on privileges to the set they actually use

mozilla LABS
Jetpack

Spring 2009: Experimental Prototype

- Firefox add-on
- a dozenish core APIs
- in-Firefox development via built-in web page
- install/remove without restarting Firefox

Fall 2009: Reboot

- SDK for professional developers w/existing toolchain
- web app for casual developers
- traditional XPI packages + APIs bundled with each add-on (akin to static linking) to bootstrap Firefox integration and enable loose coupling
- CommonJS Packages and Modules for code sharing via third-party libraries

mozilla LABS
Jetpack

Spring 2010: Debut

- initial alpha release of SDK
- preview release of web app
- project enters “incubation” phase (transition from Mozilla Labs experiment to product)

Current Status

- first beta release of SDK 1.0 on tap
  - high-level APIs (context-menu, localization, page-worker, private-browsing, request, selection, self, simple-storage, tabs, widget)
  - restartless add-ons
  - unit testing harness & dev tools
- advanced preview release of web app
  - create add-ons and API libraries
  - instant testing

What's Next?

- add-ons running in separate processes (codename: Electrolysis)
- more APIs (sidebar, password manager, panorama)
- enhancements and improvements to existing APIs and tools
- better support for CommonJS specs
- beta release of web app
- SDK 1.0 final release late Q1/early Q2 2011

Get the SDK

- download package

  v.gd/addonsdk10b1rc2

- on Windows, get Python 2.x –or– MozillaBuild

  python.org/download
  wiki.mozilla.org/MozillaBuild

- open terminal

  cmd.exe, Terminal.app, gnome-terminal, etc.

mozilla LABS
Jetpack

Activate the SDK

- expand archive

  `unzip addon-sdk-1.0b1rc2.zip`

- enter directory

  `cd addon-sdk-1.0b1`

- execute activation script

  `source bin/activate`

  – or–

  `bin\activate.bat`

Initialize Your Add-on

- change to your projects directory

  `cd ~/Projects`

- create and change to a directory for the add-on

  mkdir my-addon; cd my-addon

- execute initialization command

  cfx init

Try Your Add-on

- execute run command

  `cfx run`

- Firefox 4 beta 7 not your default browser?

```
cfx run --binary /Applications/Firefox\ 4.0b7.app/
cfx run --binary ~/firefox-4.0b7/firefox
cfx run --binary "C:\Program Files (x86)\Mozilla Firefox
4.0 Beta 7\firefox.exe"
cfx run --binary /c/Program\ Files/Mozilla\ Firefox\
4.0\ Beta\ 7/firefox
```

mozilla LABS
Jetpack

Package Your Add-on

- execute the packaging command

  `cfx xpi`

- package is written to project directory
- ready for upload to AMO!

Get Help & Participate

- SDK Documentation

  cfx docs

- Discussion Group

  groups.google.com/mozilla-labs-jetpack

- IRC Channel

  irc.mozilla.org #jetpack

- Your Presenter

  myk@mozilla.org

Going Undercover

- internal SDK files

```
packages/development-mode/
packages/test-harness/
python-lib/
static-files/
```

- getting chrome access

```
var { Components } = require("chrome");
var { Cc,Ci,Cr,Cu } = require("chrome");
```

- Electrolysis integration will impact chrome-accessing modules significantly. Stay tuned!

mozilla LABS
Jetpack

Libraries and Modules

- library: a CommonJS package that represents collection of modules implementing APIs
- module: JavaScript script that runs in its own execution context and interacts with other scripts through explicitly-defined interfaces
- Add-on Kit (stable) and API Utils (unstable) libraries come bundled with SDK
- you can create your own libraries and modules!

Create a Module

- create and open module file

```
touch lib/math.js
komodo lib/math.js
```

- export some functionality from it

```
exports.add = function(a, b) a + b;
```

mozilla LABS
Jetpack

Test the Module

- create and open test file

```
touch tests/test-math.js
komodo tests/test-math.js
```

- add a test for the exported functionality

```
exports.testAdd = function(test) {
  var math = require("math");
  test.assert(math.add(2, 3) == 5);
};
```

mozilla LABS
Jetpack

Document the Module

- create and open doc file

```
touch docs/math.md
komodo docs/math.md
```

- add documentation

```
The `math` module lets you add
numbers.  It exports one function,
`add`, that accepts two number
arguments and returns their sum.
```

- in Markdown format with some extensions

Thank you!

myk@mozilla.org

mozilla LABS
Jetpack