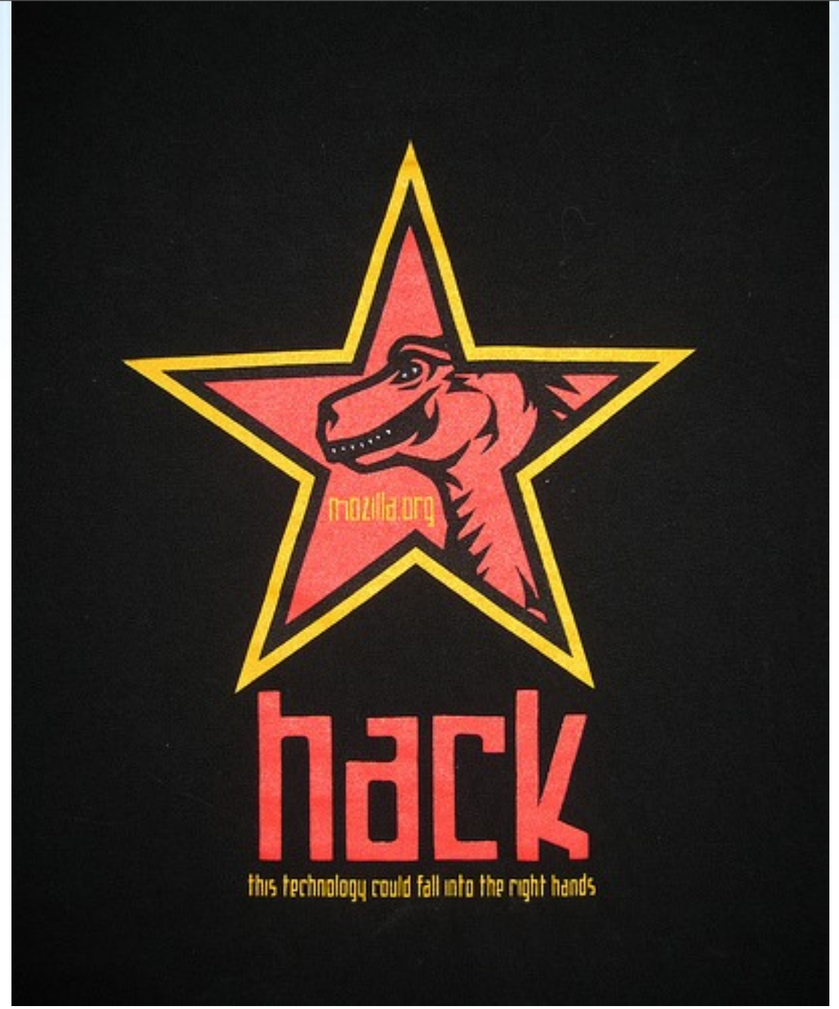
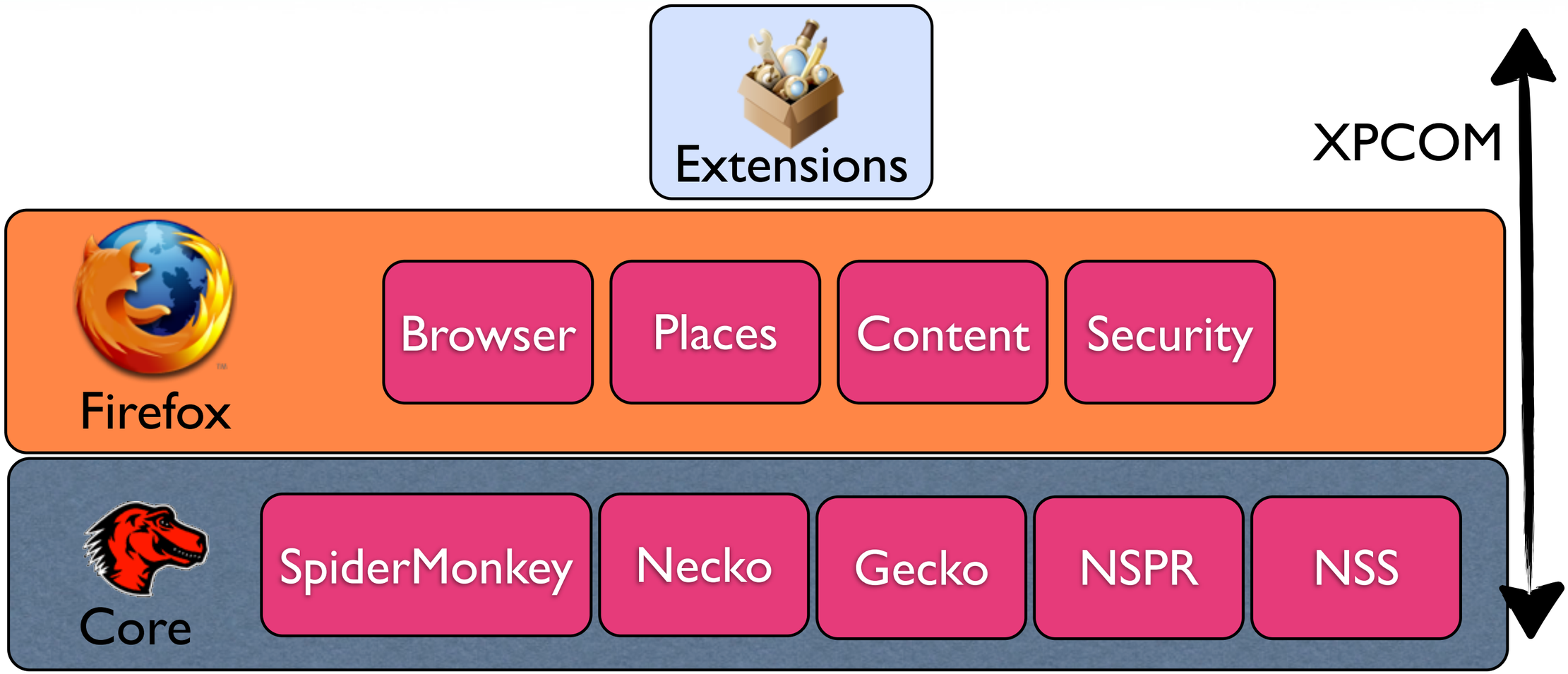


Firefox Architecture Overview

June 23, 2011

Bird's Eye View



Component Object Model

- XPCOM - The glue that binds all the pieces
  - Stable, supported bindings for C++ and JavaScript
  - Interfaces defined in dialect of IDL called XPIDL

- Any part of the code can create & use XPCOM objects
  - Core provides objects to deal with: Files, Memory, Threads, Data structures, Networking, etc.

User Interface

- XUL is our cross platform UI toolkit
- Elements are similar to HTML elements
- UI of any Mozilla-based application can be modified dynamically
  - With use of the DOM object model & common methods

2 ways to extend

 Add-ons	 Feature
Pro: Fast development time (changes can be tested by browser restart) Con: User has to explicitly install the add-on to use feature Con: Cannot change functionality not exposed by XPCOM (eg. <video>)	Con: Longer development time (requires recompile for every change) Pro: Can be rolled into the next Firefox release (every 3 months) Pro: Can change any aspect of the browser by direct code modification

Example: Read a binary file

```
const Cc = Components.classes;
const Ci = Components.interfaces;

var ios = Cc["@mozilla.org/network/io-service;1"].
          getService(Ci.nsIIOService);
var url = ios.newURI("file:///home/mozilla/test.png", null, null);

if (!url || !url.schemeIs("file")) throw "Expected a file URL.";

var pngFile = url.QueryInterface(Ci.nsIFileURL).file;
var istream = Cc["@mozilla.org/network/file-input-stream;1"].
              createInstance(Ci.nsIFileInputStream);
istream.init(pngFile, -1, -1, false);

var bstream = Cc["@mozilla.org/binaryinputstream;1"].
              createInstance(Ci.nsIBinaryInputStream);
bstream.setInputStream(istream);

var bytes = bstream.readBytes(bstream.available());
```

Example: Code path for video

Location	Interface	Purpose
/content/html/content/src/nsHTMLVideoElement.cpp	nsIHTMLVideoElement : nsIHTMLMediaElement	<video> parsed on page, object created
/netwerk/base/src/nsBaseContentStream.cpp	nsIContentStream : nsIInputStream	src='http://...' parsed and network fetch requested
/content/media/webm/nsWebMDecoder.cpp	nsWebmDecoder : nsIBuiltinDecoder	MIME type detected to be WebM, decoder object created and attached to network channel
/media/libvpx/	non Mozilla API	vp8 decoding
/layout/generic/nsVideoFrame.cpp	nsVideoFrame : nsFrame	rendering object for <video> element
/gfx/cairo/cairo/src/cairo-directfb-surface.c	non Mozilla API	low level colorspace conversion & OS-gfx painting

Example: Inject a JS API

```
const Cu = Components.utils;
const Ci = Components.interfaces;
const Cc = Components.classes;

Cu.import("resource://gre/modules/XPCOMUtils.jsm");
Cu.import("resource://gre/modules/Services.jsm");

function MyAPI() {}
MyAPI.prototype = {
    classID: Components.ID("{3d92fb7f-be77-475c-992a-5235615f9189}"),
    QueryInterface: XPCOMUtils.generateQI([
            Ci.nsIDOMGlobalPropertyInitializer,
            Ci.nsIObserver]),

    init: function MyAPI_init(aWindow) {
        let self = this;
        return {
            log: self.log.bind(self)
        };
    },

    log: function MyAPI_init(msg) {
        console.log(msg); // Privileged method
    }
};

var NSGetFactory = XPCOMUtils.generateNSGetFactory([MyAPI]);
```

JS-ctypes & restartless add-ons

- libffi binding, allows dynamic loading and execution of binary code from JavaScript

```
Components.utils.import("resource://gre/modules/ctypes.jsm");

var lib = ctypes.open("C:\\WINDOWS\\system32\\user32.dll");
var msgBox = lib.declare("MessageBoxW",
                         ctypes.winapi_abi,
                         ctypes.int32_t,
                         ctypes.int32_t,
                         ctypes.jschar.ptr,
                         ctypes.jschar.ptr,
                         ctypes.int32_t);
var MB_OK = 3;
var ret = msgBox(0, "Hello world", "title", MB_OK);

lib.close();
```

Adding functionality to Firefox is viable through both add-ons and direct feature integration, to differing extents

Injecting a JS API to web pages, interfacing with binary code, streaming data from the network and rendering video on a <canvas> element is doable as an add-on

mxr.mozilla.org
developer.mozilla.org

Questions?