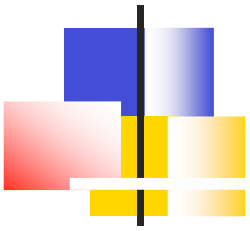
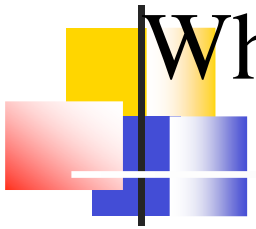


There.Is.Only.Xul

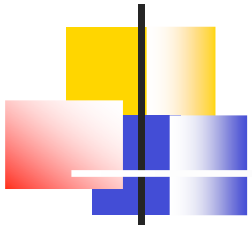
Developing RIA Applications in the Mozilla Foundation Manner



Who am I?

- Alex Olszewski
- Elucidar Software
  - Co-founder
  - Lead Developer

eLucidar



What this presentation is about?

- I was personally assigned to see how XUL and the Mozilla way measured up to RIA application development standards.
- This presentation will share my journey and ideas and hopefully open your minds to using these concepts for application development.

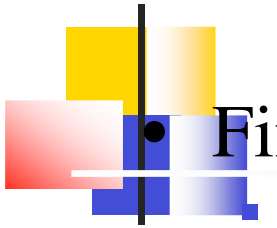
RIA and what it means

- Different to many
- “Web Applications” that have features and functions of “Desktop” applications
  - Easy install (generally requires only application install) or one-time extra(plug in)
  - Updates automatically through network connections
  - Keeps UI state on desktop and application state on server
  - Runs in a browser or known “Sandbox” environment but has ability to access native OS calls to mimic desktop applications
  - Designers can use asynchronous communication to make applications more responsive

RIA and what it means(continued)

- Success of RIA application will ultimately be measured by how will it can match user's needs, their way of thinking, and their behaviour.
- To review RIA applications take advantage of the "best" of both web and desktop apps.
- Sources:
  - http://www.keynote.com/docs/whitepapers/RichInternet_5.pdf
  - http://en.wikipedia.org/wiki/Rich_Internet_application

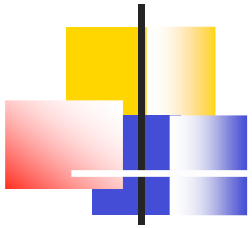
My First Steps



Find working examples

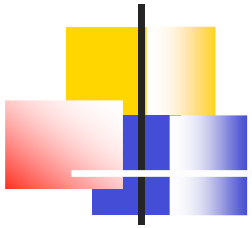
- Known Mozilla Applications
  - Firefox
  - Thunderbird
- Standalone Applications
  - Songbird
  - Joost
  - Komodo
  - FindthatFont
  - Prism (formerly webrunner) http://labs.mozilla.com/featured-projects/#prism
- XulMine-demo app http://benjamin.smedbergs.us/XULRunner/
- Mozilla Amazon Browser http://mab.mozdev.org/

What Can I do with it?



- Create Firefox browser extensions
  - Firebug, ForecasterFox,etc
- “Medium size” applications that run on “top” of Firefox, Thunderbird
  - Lightning (PIM for TB)https://addons.mozilla.org/en-US/thunderbird/addon/2313
  - Typing Drillhttps://addons.mozilla.org/en-US/firefox/addon/2309
- Create standalone applications
- ***OR ALL THREE at the SAME TIME***

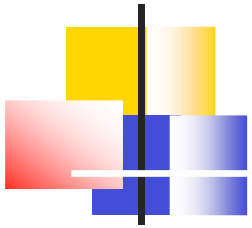
What is the Mozilla way?



- Open Standards
- Known Skill sets
- Gecko engine – Xulrunner and soon to be Firefox 3
- Build applications on proven designs for install, delivery, updates, crash reportage, internationalisation, standardization of look and feel
- Build widgets with ARIA standards (DHTML)

  Keyboard Navigation, Screen Readers ,etc
  http://developer.mozilla.org/en/docs/ARIA:_Accessible_Rich_Internet_Applications

In other words give application programmers a tested set of tools for all facets of their application lifecycle built on arguably one of the greatest Open Source Success Stories



Broad Presentation Objectives

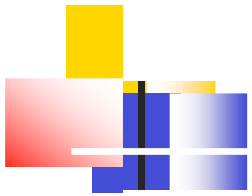
- List and Demonstrate Mozilla Application advantages
  - Application Lifecycle (Start to Finish)
  - Built on Open Source Standards
  - Leverage Existing Enterprise Skill sets
  - Moving forward
  - Leverage Firefox community and soon Firefox itself as engine.
  - Demonstrate how to build and test XUL applications.

- Demonstrate working code and what the XUL and the Gecko Engine brings to the table for you!

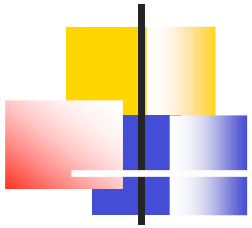
Gecko Engine

- Mozilla's Open Source Browser Engine
  - Used in various browsers
    - Firefox
    - SeaMonkey
    - Camino
- Layout Engine
  - Takes HTML, XML, image files, etc. and displays formatted content on the screen
  - Displays document's content, scroll bars, tool bars, menus, etc.

Gecko Engine

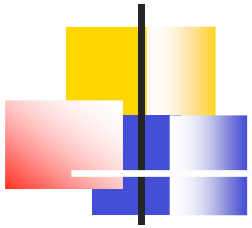


- How is this different from a browser?
  - Only provides foundation
  - Vendors may choose which components to package
- Was designed and implemented to allow Mozilla developers a UI framework for applications(The Firefox browser to start with) such that:
  - Modular
  - Abstracted away OS specific layout and control widgets
  - Allowed for extension points from the widgets and native processes i.e. File IO, networking, etc



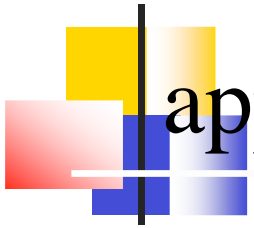
Gecko Languages/Technologies

- XUL UI Elements – namespace XUL
  - Window
  - Hbox, Vbox, dialog, tree, wizardpage,
  - Grids, Columns,
  - Stacks,Decks
  - Textboxes, Richtextboxes,buttons
  - Many widgets to build an app
  - http://www.hevanet.com/acorbin/xul/top.xul (Xul Periodical Table)
  - http://www.xulplanet.com/references/elemref/ (Xul Planet reference)
- XPCom – Cross Platform Bindings
- XBL Bindings-XML Binding Language or Extensible Binding Language
- JavaScript
- CSS
- DOM ver2
- Many more…(Really anything that the Gecko Layout Engine understands)



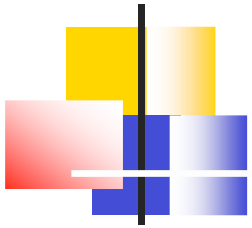
Xul UI

- Composed of Elements (Widgets)
- Can be HTML
- SVG –Scalable Vector Graphics
- XForms
- And really inclusion of other XML namespace aware data see Venice Project (Joost) and <sprite> (to be soon donated to Mozilla



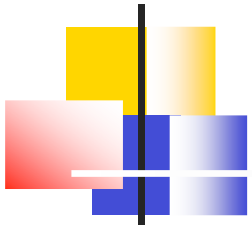
Testing and Developing XUL applications(before further presentation)

- Editing
- Viewing
- Directory Layout
- Debugging



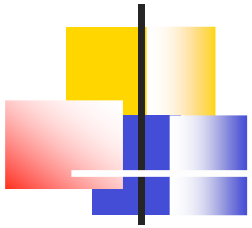
Editing

- Use any text editor that is “XML” friendly.
- Komodo (written in XUL , IDE, EDIT)
  - IDE is proprietary
  - Edit is Open Source http://www.openkomodo.com/
- Eclipse, JetBrains,
- EditPlus, ProgrammersNotepad, etc
- TextPad



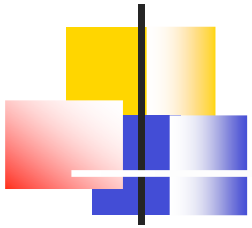
Viewing

- Using Firefox version x
  - When using version 2 –client will cause it to run
  - When using version 3 -app
- Using XULRunner is the standalone portion of gecko runtime
- Deploying application under chrome directory of Firefox x on system
- Pluses and minuses
  - When in production lifecycle – my hope (still mozpad groups are not in full agreement) is that Firefox and XULRunner are same and therefore update issue of “Engine” is partially solved.
  - When deploying under system Firefox browser, debugging is easier, but same things are accomplishable under XULRunner with configuration.



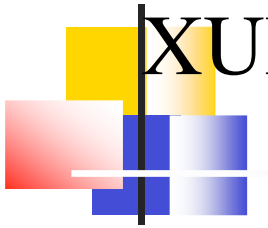
Viewing (continued)

- To *view* 2 ways:
  - Put directories under chrome directory in ~Program files/Firefox:
  - Put on File system and run using XULRunner or Firefox 3 with slightly different directory structure
  - Which leads us to XUL directory structure:



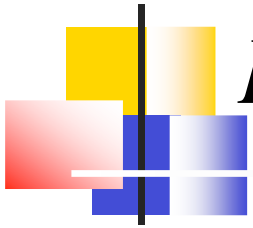
XUL Structure

- What accomplishes many goals
- IE. Localization, Installation, Update mechanism, etc
- URL management , start to think of it as "*similar*" to virtual directories that are "mapped"



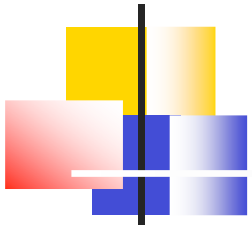
XUL Structure (continued, simple version)

- ApplicationContainmentDir (MyApp)
  - Contains NON-Mozilla files (project files example eclipse .proj file)
    - But also contains application .ini
  - Chrome – could be thought of as a virtual dir…
    - *It is mapped differently later depending upon whether your application runs as an extension or stand-alone application*
    - *It also has it's own URLs which are thought of interface elements OUTSIDE the window's content area.. (MORE ON CHROME LATER)..*
    - http://developer.mozilla.org/en/docs/Chrome_Registration#What_is_Chrome.3F
  - Defaults



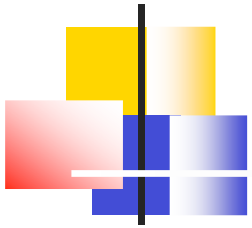
Chrome Directory-will map as *"Chrome Providers"*

- Content – contains layout, .js and bindings for now.
- Skin- will contain "Application specific" Look and feel, CSS and images
- Locale – will contain values that can be replaced pertaining to your system. "Internationalisation"



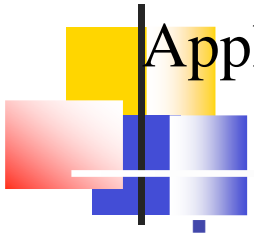
Simple Demo-Let's Get Started..

- Will show a simple window with XUL widgets for really simple layout,
- Then include html namespace to add HTML
- Then include SVG namespace to include simple SVG.
- Purpose of all this is to get familiar with testing, show errors, and basic "Package" structure of Mozilla application



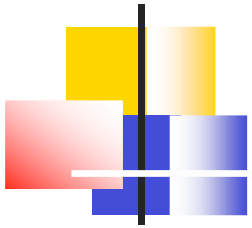
Ok what is CHROME?

- UI around the web page (as opposed to content area).
- As you will see chrome://app/ urls and has privileges
- Passing the chrome:// url to window.open will open the application *"Without a Browser"*
- Folder where the content and often the client side application exists
- When we get into mapping chrome URL provides way to identify "*Chrome Providers*" ..
  - ***Courtesy..***
    - http://developer.Mozilla.org/en/docs/Chrome



Application.ini

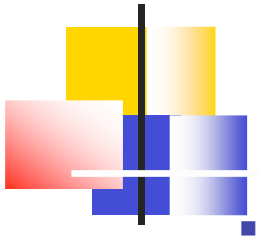
- [App]http://developer.mozilla.org/en/docs/XUL_Application_Packaging
  - Name (required)
  - Version(required)http://developer.mozilla.org/en/docs/Toolkit_version_format
  - ID (required)email or UUID
  - BuildID(required) timestamp
  - Vendor(optional)
  - Profile(optional)
- [Gecko] what version of XULRunner is required by the application
  - MinVersion
  - MaxVersion
- [XRE] various features of XULRunner startup that can be enabled
  - EnableExtensionManager (optional)
  - EnableProfileMigrator (optional
- [Crash Reporter]http://code.google.com/p/socorro/
  - Enabled=True
  - ServerURL=https://your.server.url/submit



XUL Layouts

- All Xul UI elements have common attributes and properties
- Many affect their layout
- Many affect their children's layout. Some are:
  - Height, width, top, flex, max(height,width), min(height,width), style, align, left, pack, position, equalsize, persist, etc.

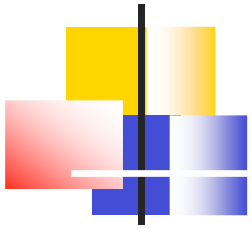
- Two places that are good for reference:
  - http://developer.mozilla.org/en/docs/Sandbox:XUL_element_format
  - http://www.xulplanet.com/references/elemref/ref_XULElement.html



XUL and the Box Model

- Top Box - “Window”
  - And window “*Like*”
    - Dialog,
    - Wizard (and WizardPages)
    - Overlay

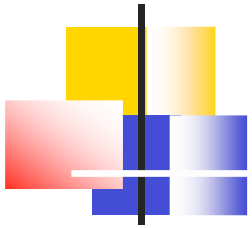
- Grouping boxes (all about orientation)
  - Box is a(see below) by orient property default Horizontal
  - Hbox -horizontal alignment
  - Vbox –vertical alignment
  - GroupBox groups objects
  - Grid
  - Menu (with menubars, etc)



Box Model Layout-Sizing

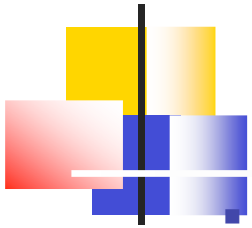
- Boxes (appearance aside) will generally size to what they “need” to be , i.e. what their children need to be if they can.
- An element will be as large as it needs to be and no larger by default.” Hold it’s contents”
- Children position within their containers based on orientation layout , and computed size. To help give more control:
  - Spacer elements
  - Use Min Max sizes
  - Use flex(any value greater than 0) grow and fit available space by flex value..
  - Start, center, end positioning values
  - Stretch (like flex but opposite direction , based on box orientation
  - Many of these values are available in CSS even if you need to use Mozilla CSS custom extension http://developer.mozilla.org/en/docs/CSS_Reference:Mozilla_Extensions
  - * Example App http://www.xulplanet.com/tutorials/xultu/examples/aligner.xul
- Children position within their “container” based upon the alignment, their computed size,and their flexibility

XULedit Demo

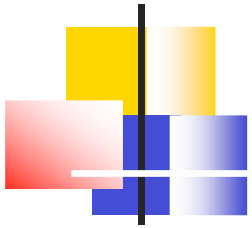


- XUL app which lets you rapidly prototype XUL
- http://ted.mielczarek.org/code/mozilla/xuledit/index.html
- Allows you to work out issues quickly
- Is downloadable as standalone
- Is part of Extension Library for Firefox (next slide)

Debugging

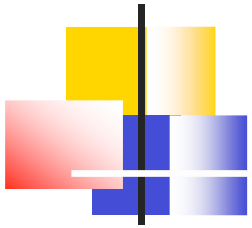


- ***Depending on where application lives on File System***
- Under Firefox chrome directory
  - Use normal debug tools
  - Download Chrome List extension.https://addons.mozilla.org/en-US/firefox/addon/4453
  - Xul Reference https://addons.mozilla.org/en-US/firefox/addon/3953
  - Extension Developers Extension *http://ted.mielczarek.org/code/mozilla/extensiondev/



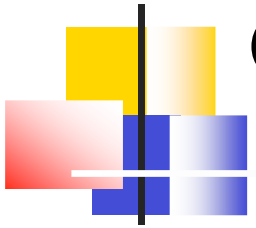
Debugging (continued)

- Anyplace else using XulRunner or –app or –client
  http://developer.mozilla.org/en/docs/Debugging_a_XULRunner_Application - Prefs
  - Add to prefs (could be default,or debug).js following:
  - /* debugging prefs */ pref("browser.dom.window.dump.enabled", true); pref("javascript.options.showInConsole", true);
  - pref("javascript.options.strict", true);
  - pref("nglayout.debug.disable_xul_cache", true);
  - pref("nglayout.debug.disable_xul_fastload", true);
  - Start XULRunner with –console (after path to application.ini)
  - Can use then use dump('foo') instead of alerts..



Debugging(continued)

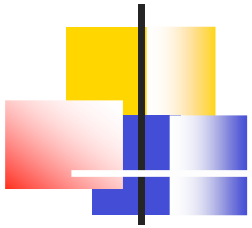
- And/Or for JS errors –jsconsole
  - Can use Components.utils.reportError(string)
  - Or view
    - http://developer.mozilla.org/en/docs/JavaScript_Console
    - http://developer.mozilla.org/en/docs/Console_service
- Can even add Venkman JS debugger
  - Read tutorial http://developer.mozilla.org/en/docs/Debugging_a_XULRunner_Application - Venkman
- ***Key is to Be Flexible to Your debugging needs of the Moment!***



Code- Let's Make it DO Something

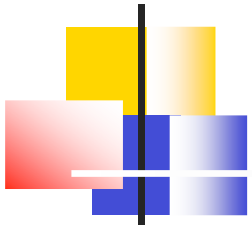
- As noted before Could just attach JavaScript to Elements (use events)
- XUL is part of the DOM
- Could have HTML elements or XUL equivalents
- Have known set of manipulation methods
- From XUL element:

- blur, click, doCommand, focus, getElementsByAttribute
- Inherited from DOM element
  addEventListener, appendChild, dispatchEvent, getAttribute, getAttributeNode, getAttributeNodeNS, getAttributeNS, getElementsByTagName, getElementsByTagNameNS, hasAttribute, hasAttributeNS, hasAttributes, hasChildNodes, insertBefore, isSupported, normalize, removeAttribute, removeAttributeNode, removeAttributeNS, removeChild, removeEventListener, replaceChild, setAttribute, setAttributeNode, setAttributeNodeNS



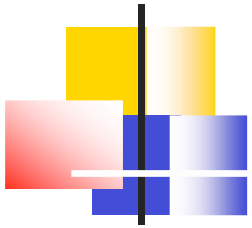
XUL JavaScript Demo

- Manipulate DOM elements
- Handle Events
- All demo code will be available on post event site



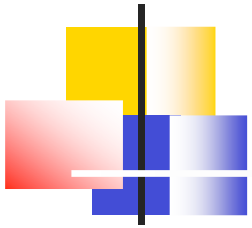
What this means

- Whatever I wanted to do from a “Browser” , I can still do with the ability to build my own interface widget it I wanted to , BUT YOU PROMISED ME MORE!
- That’s where the Binding libraries come in, but last thought before we leave this is that yes you can use “some” JS libraries, to a degree, i.e JQuery, just not to do all things, it was good for “*SELECTION*” , not so good for addition of elements, they promise more support in the future.
- Bear Bibeault is an expert on JQuery, Prototype, and Scriptaculous and is a speaker here. He may have some insight on XUL and XML support in these libraries.



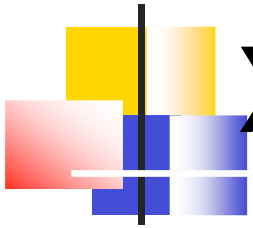
XPCom-What is IT?

- X(Cross)Platform Component Object Model
- Allows for Objects to be expressed as Interfaces using an IDL (Interface Description Language) known as XPIDL,it is a tool (compiler) that can generate
  - C++ header files
  - java interface files
  - (using)XPConnect typelib (.xpt) files used to dynamically bridge objects for JavaScript and implement the interface in JavaScript thus bridge C++ to scripting languages



XPCom –What is IT? continued

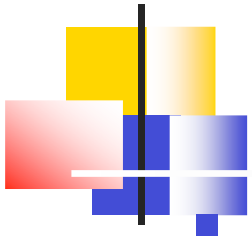
- There are similar libraries (language bindings) for:
  - Ruby(RbXpCom)
  - Python(PyXpCom)
  - Perl(PlXpCom)
- But before we even worry about that there are over 1300 interfaces in the Mozilla tree alone, let alone other open source applications. So let's look at how to use them.



XPCom

- Using an implemented component
  - Use “Aware” project
  - Use “nsIFileLocal” demo

- Writing a new XPCom interface and implementing it is large subject and can not be covered in this time period.
- If you are interested full coverage ,and with a 5 part IBM series, look Here
- Also as an introductory app, Here is a “Hello World” XPCom example for JavaScript, that has a wrapper and does not need the compiler.



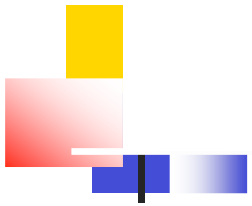
Installers

- Set up your directory structures accordingly!
- Standalone
  - will for now have XulRunner dir, defaults dir(preferences!) and application.ini.
  - XulRunner stub comes with XulRunner rename to your application ..(starts XULRunner) easiest is to review:
  - http://developer.mozilla.org/en/docs/How_to_Create_Windows_Inno_Setup_Installer_for_XULRunner_Application
  - As a add-on or extension (.xpi file) Directory structure is the same (there are even build scripts available for all platforms)

Overlays

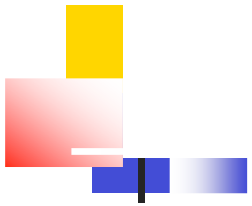
- Provide a mechanism for
  - Adding UI for additional components
  - Overriding small pieces of a XUL file without having to resupply the whole UI
  - Reusing pieces of the UI
- 2 Ways to include overlay
  - In XUL document
    - <?xul-overlay href="myOverlay.xul"?>
  - In manifest
    - overlay chrome://global/content/source.xul chrome://browser/content/overlay.xul

Overlay Demo



- Modify Menu Elements through Overlays
- Again all demo code available later

XBL



- Extensible Bindings Language
  - Describes bindings that can be attached to elements in other documents
    - XUL
      - Customize look with styles
      - Modify appearance with element attributes
      - No way to change how an element works
  - bound element
    - element that the binding is attached to
    - acquires the new behaviour specified by the binding

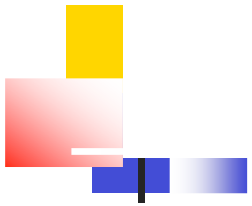
XBL Demo

- Modify <box> element

il8n

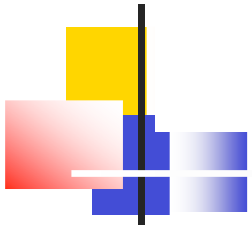
- DTD support
  - Used in XML
  - <button label="&button.ok;" />
- String Bundle
  - Accessed via JavaScript
    - demoResourceBundle.getString("button.ok");

il8n Demo

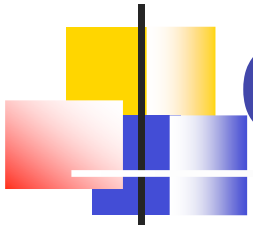


- DTD support
- String Bundle

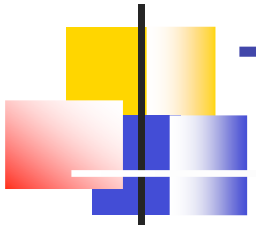
Aware Demo



- Hide Chrome
- Open Multiple Windows
- Xml Templates



Questions?



Thank You!