Tooling for Ajax-Based Development

Craig R. McClanahan
Senior Staff Engineer
Sun Microsystems, Inc.
In The Beginning
In The Beginning
How Did We Start Writing Ajax Applications?

• Hand-coded XMLHttpRequest (XHR) calls
  > Tedious code
  > Browser-specific quirks
  > Limited reusability
In The Beginning

How Did We Start Writing Ajax Applications?

```javascript
function ajax($url, $vars, $object) {
    if (window.XMLHttpRequest) {
        var $class = new XMLHttpRequest();
    } else { var $class =
             new ActiveXObject("MSXML2.XMLHTTP.3.0");
    }
    $class.open("POST", $url, true);
    $class.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
    $class.onreadystatechange = function() {
        if ($class.readyState==4 && $class.status==200) {
            if ($class.responseText) {
                $obj = $class.responseText; $object($obj);
            }
        }
    }
    $class.send($vars);
}
```
In The Beginning

How Did We Start Writing Ajax Applications?

• Hand-coded Dynamic HTML (DHTML) manipulation
  > Tedious code
  > Browser-specific quirks
  > Limited reusability
In The Beginning
How Did We Start Writing Ajax Applications?

```javascript
component = Document.getElementById("options");
component._clear();
component.menu.style.visibility = "visible";
var children = null;
var items = data.getElementsByTagName("item");
if ((items != null) && (items.length > 0)) {
    for (loop = 0; loop < items.length; loop++) {
        children = items[loop].childNodes;
        if ((children != null) && (children.length > 0)) {
            component._append(children[0].nodeValue);
        }
    }
}
```
In The Beginning
How Could We Do A Better Job?
Frameworks ... lots of frameworks!
Frameworks
Frameworks

Client Side Frameworks

• Encapsulate common behaviors:
  > Asynchronous communication to server
  > Dynamic HTML widgets
  > Client side events
  > Dynamic visual effects

• Examples:
  > Dojo, Prototype, Scriptaculous, Jquery, Yahoo! UI ...
Frameworks

Client Side Frameworks

• But can I use them together?
• Open Ajax Alliance

> Goal – Accelerate success with Ajax by promoting mix and match solutions based on interoperable technologies
> Many of the organizations presenting here are members
Frameworks

Client Side Frameworks

• But can I use them together?
• The jMaki Framework
  > https://ajax.dev.java.net/
  > Lightweight framework for constructing JavaScript centric Web 2.0 applications using CSS layouts, a widget model, cross library data models, and cross library eventing
  > Choice of server side runtimes:
    – Java/JSP, Java/JSF, PHP, Server Side JavaScript (Phobos), and Ruby on Rails
  > Plugins for NetBeans and Eclipse
Frameworks

Server Side Frameworks

• Often supported by updated versions of traditional web application frameworks:
  > Java – Struts, JSF, Tapestry, Wicket, Stripes, ...
  > PHP – Zend, CakePHP, Symfony, Seagull, ...
  > Python – Zope, Django, Webware, SkunkWeb, ...
  > Ruby – Rails, Nitro, Iowa, ...

• Two general approaches:
  > Markup generated server side (including Ajax callbacks)
  > Widget calls generated server side, markup generated client side (data exchanged via Ajax callbacks)
Frameworks

End To End Frameworks

• Google Web Toolkit:
  > Write and debug client side code in Java
  > Translated into JavaScript for download to browser
  > RPC mechanism for remote data transfer

• ICEFaces:
  > JavaServer Faces component library, so applications developed in Java
  > Incremental DOM updates delivered dynamically via Ajax
  > Ajax Push – Server initiated content rendering
Frameworks

End To End Frameworks

• **Google Web Toolkit:**
  > Write and debug client side code in Java
  > Translated into JavaScript for download to browser
  > RPC mechanism for remote data transfer

• **ICEFaces:**
  > Based on a JavaServer Faces component library, so applications developed in Java
  > Incremental DOM updates delivered dynamically via Ajax
  > Ajax Push – Server initiated content rendering
Tooling
Tooling

Why Do Tools Matter?

• Traditionalists like **emacs** and **vi** for development
  > Why? Because they are **productive** with these tools
  > Personal history – long time emacs user – mouse hater :-)

• Can a GUI or IDE environment improve productivity?
  > If it cannot, *stay away from them*
  > If it can, *pay attention to them*
Tooling

I'm a Power Developer ... How Can IDEs Help Me?

- Color coded editing – nice but already available
- Code completion
- Documentation lookup
- Find usages
- Refactoring
- GUI editors for configuration files
- Debugging
- Code generation wizards
Tooling

Tooling Demos

• Based on NetBeans 6.0 Milestone 10:
  > Similar capabilities available in Eclipse, IntelliJ, ...

• Client side use cases:
  > Google Web Toolkit
  > jMaki with JavaServer Pages
  > jMaki with JavaServer Faces
  > jMaki with Ruby on Rails

• Server side use cases:
  > Java Persistence API entity class generation
  > RESTful web service from entity classes
Architecture Approaches
Architecture Approaches

Different Architectural Styles

• Most Ajax-oriented presentations concentrate on:
  > Client side libraries
  > End-to-end solutions

• How do we deal with adding Ajax to existing apps?

• How do we architect for next generation web scale?

• Consider the following styles:
  > Traditional server side web application with eye candy
  > Traditional webapp with partial page submit/refresh
  > All In One Page (AIOP)
  > Non JS+DHTML Solutions
Architecture Approaches

Traditional Server Side With Eye Candy

• Millions of currently deployed applications that are:
  > Based on traditional web application frameworks
  > Based on traditional HTTP page semantics
  > Could offer usability improvements from simple Ajax interactions

• Look for opportunities to leverage:
  > Client side widget libraries directly
  > Low hanging fruit – animations that would be unfeasible with full page refreshes:
    – Progress bars, animated voting tools, autocomplete text fields, ...
  > Background Ajax to newly created URI endpoints
Architecture Approaches

Partial Page Submit / Partial Page Refresh

• Start with a traditional web application
• Divide page content into separate areas that can (logically) be submitted and/or refreshed separately
• Continue to leverage server side markup creation
• Example implementation technologies:
  > Ruby on Rails – Partials
  > JSF – Project Dynamic Faces
Architecture Approaches

All In One Page (AIOP)

• Best suited to green field development
• Move view and controller to client side
• Use Ajax requests to exchange *data* not *markup*
  > Reuse *simple web services* exposing your content and functionality for:
    – Internal HTML based UI
    – Internal AIOP based clients
    – External mashups

• Need some sort of MVC framework on client side
• Side note – give up on traditional web analytics :-)


Architecture Approaches

Non JS+DHTML Solutions

• Ajax (implemented with JS+DHTML) is not the only possible solution for rich client UIs

• Browser plugins offer competing solutions:
  > Flash, Flex, Silverlight, Java FX

• So do traditionally installed client applications:
  > Nowadays, better known as Rich Internet Applications

• If we want Ajax to succeed, we need:
  > Higher quality JS+DHTML implementations in browsers
  > Better interoperability between implementations
  > Comprehensive solution to code injections with eval()
Resources
Resources

Additional Resource Pointers

• NetBeans 6.0 Milestone 10:
  > http://www.netbeans.org/community/releases/60/index.html

• Sun Web Developer Pack:
  > http://developers.sun.com/web/swdp/

• jMaki On Rails Plugins Repository (alpha quality):
  > http://jmaki-goodies.googlecode.com/svn/trunk/rails_plugins/

• Project Dynamic Faces:
  > https://jsf-extensions.dev.java.net/
Q & A
Tooling for Ajax-Based Development

Craig R. McClanahan
Craig.McClanahan@sun.com