



AJAX Toolkits and Frameworks

Sang Shin
Java Technology Architect
Sun Microsystems, Inc.
sang.shin@sun.com
www.javapassion.com

Disclaimer & Acknowledgments

- Even though Sang Shin is a full-time employee of Sun Microsystems, the contents here are created as his own personal endeavor and thus does not necessarily reflect any official stance of Sun Microsystems on any particular technology

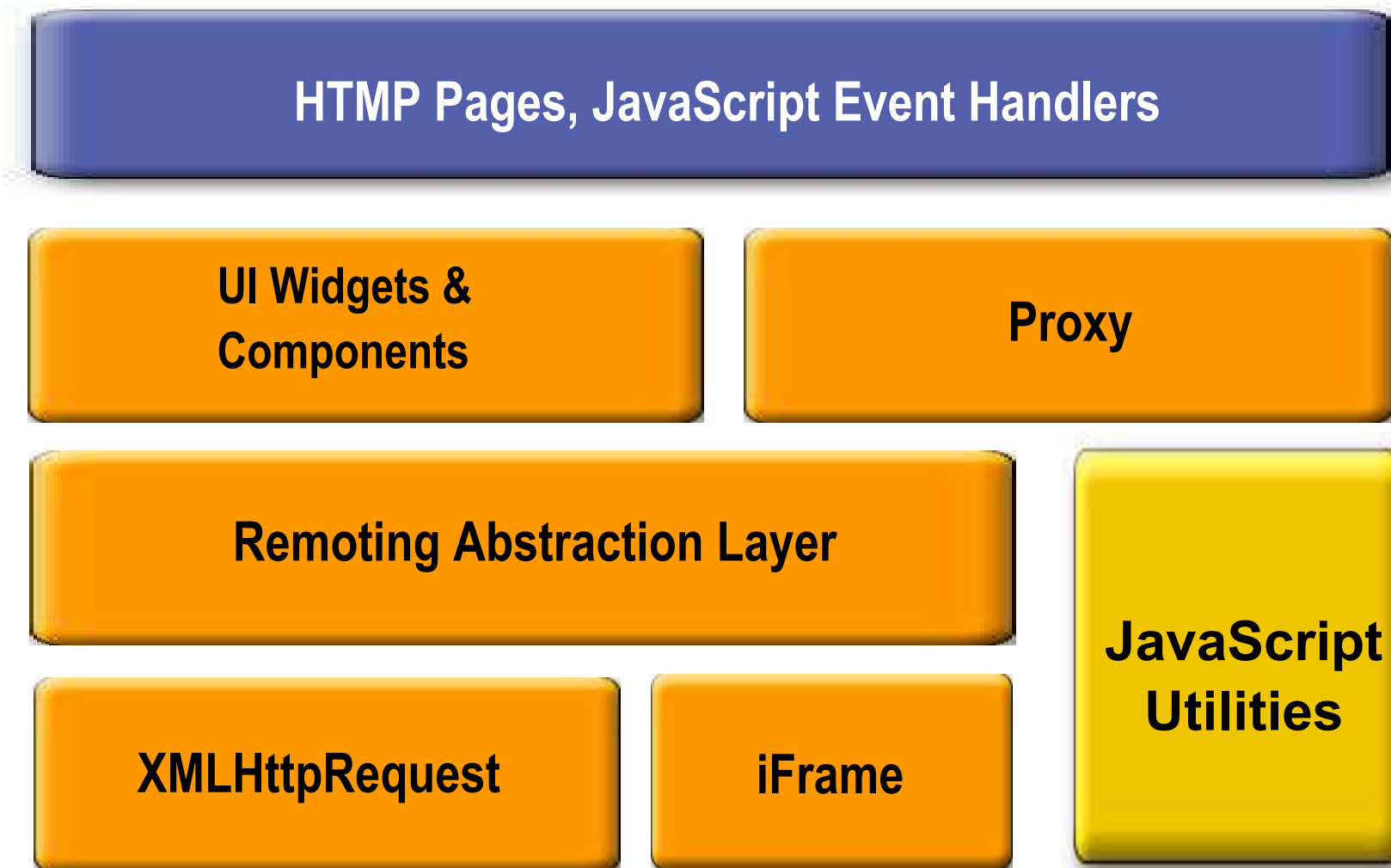
The Goal of This Presentation

- Just to give you a sense of what types of AJAX toolkit and framework solutions are available out there today
 - > Each of these will be talked about in detail in other presentations
- Give you some rough guidance on which technology to use under what circumstances
 - > This is based on Sang Shin's personal observations: Others might have different views
 - > You will be the ultimate judge based on your needs

Types of AJAX Toolkit and Framework Solutions of Today

- Clients-side JavaScript Libraries
- RMI-like remoting via proxy
- AJAX-enabled JSF components
- Wrapper (jMaki)
- Java to JavaScript/HTML translator (GWT)
- Web Application Frameworks with AJAX extension
- A few more out there

Architectural Layers (Client-Side)



Architectural Layers (Client-side)

- Remoting abstraction layer
 - > Hides handling of XMLHttpRequest and IFrame
- Proxy
 - > Handles client side of RPC like communication
- Widgets and components
 - > Provides ready-to-use UI widgets such as calendar, button, etc
- JavaScript event handlers
 - > Provides client-side logic

Client Side JavaScript Libraries

Client Side JavaScript Libraries

HTMP Pages, JavaScript Event Handlers

UI Widgets &
Components

Remoting Abstraction Layer

XMLHttpRequest

iFrame

JavaScript
Utilities

Characteristics of Client Side JavaScript Libraries

- Server side technology agnostic
 - > The server side technology can be Java EE, .Net, PHP, Ruby on Rails, etc.
- Should be accessible during runtime either locally or through URL
 - > There is no dynamic JavaScript code generation
- You can use them in combination in a single app
 - > You might want to use widgets and JavaScript utilities from multiple sources

Technical Reasons for using Client-side JavaScript Libraries

- Handles remote asynch. communication (remoting)
 - > Hides low-level XMLHttpRequest operation
- Handles browser incompatibilities
 - > No need to clutter your code with if/else's
- Handles graceful degradation
 - > Uses **IFrame** if the browser does not support XMLHttpRequest
- Provides page navigation hooks over AJAX
 - > Back and forward buttons
 - > Bookmarking

Technical Reasons for using Client-side JavaScript Libraries

- Provides ready-to-use widgets
 - > Tree, Calendar, Textfield, Button, Split panes, Fisheye, etc.
- Provides easy-to-use DOM utility
 - > Easier to use than original DOM APIs
- Provides useful JavaScript utilities
 - > Example: Table management, Timer, etc
- Provides error handling hook
 - > Easier to add error handler
- Provides more flexible event handling
 - > DOM node based, Function call based, AOP style

Technical Reasons for using Client-side JavaScript Libraries

- Provides advanced UI features
 - > Animation
 - > Drag and drop
 - > Fade out and Fade in
- Generally encourages OO programming style
 - > Helps you write better JavaScript code

Business Reasons for using Client-side JavaScript Libraries

- Proven in the market
 - > Generally higher quality than your own
- Established developer/user communities
 - > Community keeps improving/adding features
 - > Easy to get help from community forums
- Easy to use
 - > It is just a matter of having them in the root directory of your Web application or providing URL location
- Tool support
 - > IDE's will support them in time

Client-side JavaScript Libraries

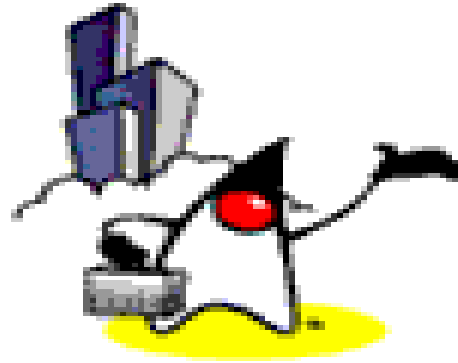
- DOJO Toolkit
 - > Most prominent and comprehensive
 - > Gaining a leadership in this space
 - > Major industry support (Sun, IBM)
 - > <http://dojotoolkit.com/>
- Prototype
 - > Used by other toolkit libraries
 - > <http://prototype.conio.net/>

Client-side JavaScript Libraries

- Script.aculo.us
 - > Built on Prototype
 - > Nice set of visual effects and controls
 - > <http://script.aculo.us/>
- Rico
 - > Built on Prototype
 - > Rich AJAX components and effects
 - > <http://openrico.org/>
- DHTML Goodies
 - > Various DHTML and AJAX scripts
 - > <http://www.dhtmlgoodies.com/>

Pro's And Con's

- Pro's
 - > You can use it with any server side technology
 - > Many widgets from multiple sources
- Con's
 - > Developer still has to learn JavaScript
 - > Different libraries use different syntax
- When to use
 - > You need to work with multiple server side technologies
 - > You want to use widgets from multiple sources (jMaki will help here assuming you are using Java EE)



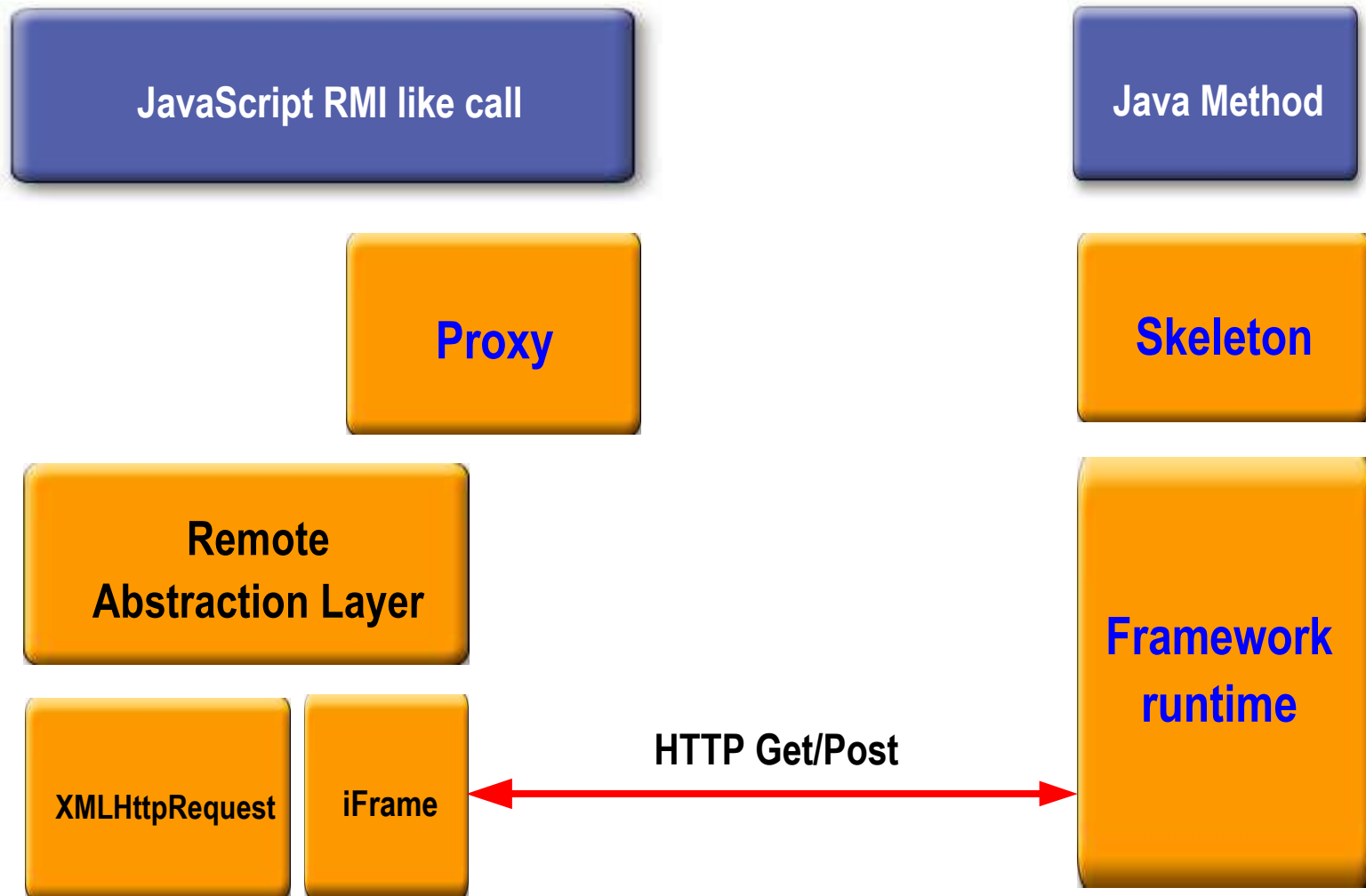
Demo: Running Widgets from Dojo, Script.aculo.us, Rico, DHTML Goodies

Demo Scenario: Run Online Demos

- Dojo
 - > <http://archive.dojotoolkit.org/nightly/demos/widget>
- Script.aculo.us
 - > <http://wiki.script.aculo.us/scriptaculous/show/Demos>
- Rico
 - > <http://openrico.org/rico/demos.page>
- DHTML Goodies
 - > <http://www.dhtmlgoodies.com/>

RMI-like Remoting via Proxy

RMI-like Remoting via Proxy



Characteristics of “RMI-like Remoting via Proxy” Framework

- Similar to general RPC communication schemes
 - > Stub and Skeleton based architecture
- Allows RMI like syntax in the client side JavaScript code
- Framework generates client stub (Proxy), which is a JavaScript code
- Framework provides server side runtime as well
- Client stub (Proxy) handles marshalling of parameters and return value

Remoting via Proxy Implementations

- Direct Web Remoting (DWR)
 - > Designed specifically for Java application at the backend
 - > <http://getahead.ltd.uk/dwr>
- JSON-RPC
 - > Lightweight remote procedure call protocol similar to XML-RPC
 - > <http://json-rpc.org/>
 - > There are language-specific implementations
 - > JSON-RPC-Java
 - > <http://oss.metaparadigm.com/jsonrpc/>

Pro's And Con's

- Pro's
 - > Allows you to expose existing backend business logic to AJAX client with minimum work
 - > Allows you to use familiar RMI like syntax in the client side JavaScript code
- Con's
 - > Hackers can see what backend methods are available
 - > Custom converter (marshaller and unmarshaller) needs to be created for custom Java objects
- When to use
 - > When you want to expose existing backend business logic to AJAX client with minimum effort

AJAX-Enabled JSF Components

AJAX-enabled JSF Components

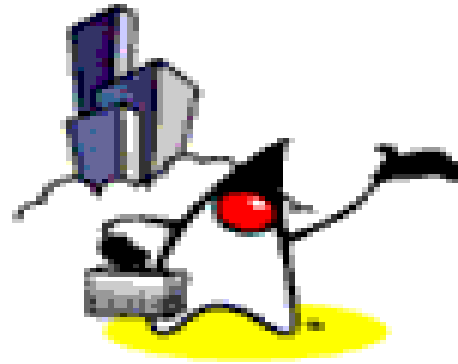
- AJAX-enabled JSF components hides all the complexity of AJAX programming
 - > Page author does not need to know JavaScript
 - > The burden is shifted to component developers
- Leverages drag-and-drop Web application development model of JSF through an IDE
 - > You can drag and drop AJAX-enabled JSF components within Sun Java Studio Creator 2 (and other JSF-aware IDE's) to build AJAX applications
- JSF components are reusable
 - > More AJAX-enabled JSF components are being built by the community

Implementations

- Blueprint AJAX-enabled JSF components (open-source)
 - > <http://developers.sun.com/ajax/componentscatalog.jsp>
 - > <https://bpcatalog.dev.java.net/ajax/jsf-ajax/>
- ajax4jsf (open-source)
 - > Can add AJAX capability to existing applications
 - > <https://ajax4jsf.dev.java.net/>
- ICEfaces (ICESoft) - commercial
 - > <http://www.icesoft.com/products/icefaces.html>
- DynaFaces (development on-going)
 - > <https://jsf-extensions.dev.java.net/nonav/mvn/slides.html>

Pro's And Con's

- Pro's
 - > Drag-and-dropping AJAX-enabled JSF components within an IDE for building AJAX application
 - > Leveraging 3rd-party AJAX-enabled JSF components
- Con's
 - > Building your own AJAX-enabled JSF component is not trivial task
- When to use
 - > When you want to build AJAX apps by drag-and-dropp'ing
 - > When you are already committed to JSF programming model for building your applications (especially using an JSF-aware IDE)
 - > When you want to avoid JavaScript coding



Demo: Running Demos from various JSF/AJAX Solutions

Demo Scenario

- Blueprint AJAX-enabled JSF components
 - > <http://developers.sun.com/ajax/componentscatalog.jsp>
- ajax4jsf
 - > <http://livedemo.exadel.com/vcpDemo/demo.jsf> (commercial version based on open source ajax4jsf)
 - > Changing skins, Drag & Drop
- ICEFaces
 - > http://www.icesoft.com/products/demos_icefaces.html

Wrapper Technology: jMaki

Motivations for jMaki

- You want to leverage widgets from existing and future AJAX toolkits and frameworks
 - > Dojo, Scriptaculus, Yahoo UI Widgets and DHTML Goodies
- Today, there are multiple AJAX frameworks with their own widgets and with different syntax
 - > There is a need for a common programming model for using widgets from multiple AJAX toolkits and frameworks
- JavaScript coding is too alien to many Java EE developers
 - > There is a need for using JavaScript widgets using Java EE syntax and programming model

What is jMaki?

- JavaScript Wrapper framework for the Java platform
 - > The name, jMaki, was derived from "j," for Java, and "Maki," a Japanese word for wrap
- Allows developers to take widgets from many popular AJAX toolkits and frameworks, and wrap them into a JSP or JSF tag
 - > Provides a common programming model to developers
 - > Leverages the widgets from popular frameworks
 - > JSP and JSF tags are familiar to Java EE application developers

Wrapper Technology Implementations

- jMaki
 - > <https://ajax.dev.java.net/>

Pro's And Con's

- Pro's
 - > Provides unified programming model for using widgets over various AJAX toolkits and frameworks
 - > Allows Java developers to use familiar Java EE programming model (JSP or JSF tags) for using JavaScript widgets
 - > There is already a NetBeans Plug-in (for NetBeans 5.5)
- Con's
 - > Event model is still not fully baked yet
- When to use
 - > When you want to use widgets from different sources yet want to use uniform programming model

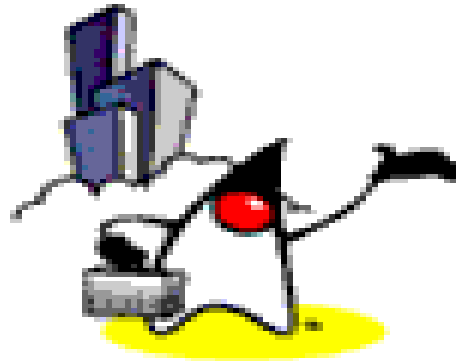
Java Code To JavaScript/HTML Translator: GWT

Java to JavaScript/HTML Translator

- Develop and debug AJAX applications in the Java language using the Java development tools of your choice
- When you deploy your application to production, the GWT compiler translates your Java application to browser-compliant JavaScript and HTML

Pro's And Con's

- Pro's
 - > Allows Java developers to use Java language for the development AJAX applications including debugging/testing
 - > No need to learn JavaScript language
 - > It is from Google (Good momentum + Good document + Good tutorials + Comprehensive)
- Con's
 - > (If I have to find a con), you lose control since the translator does all the work for you
- When to use
 - > When you already have Swing expertise



Demo: Running GWT Online Demos

Demo Scenario

- GWT Kitchen Sink demo
 - > <http://code.google.com/webtoolkit/documentation/examples/kitchensink/demo.html>
- GWT Dynamic Table example demo
 - > <http://code.google.com/webtoolkit/documentation/examples/dynamictable/demo.html>

Web Application Frameworks with AJAX Extension

Web App. Framework with AJAX Extension

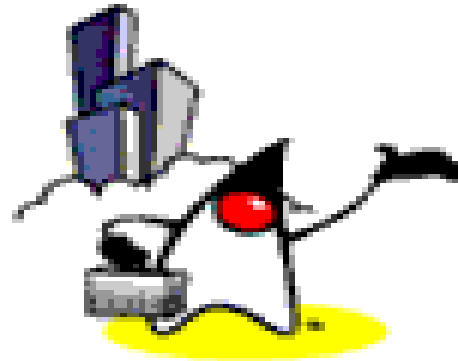
- Existing Web Application Frameworks add AJAX functionality
 - > Minimum or no requirement of JavaScript coding
- Uses JavaScript client library internally

“Web App Frameworks with AJAX Extension” Implementations

- Wicket
 - > <http://wicket.sourceforge.net/>
- Echo2
 - > <http://www.nextapp.com/platform/echo2/echo/>
- Shale
 - > <http://struts.apache.org/struts-shale/>
- Ruby on Rails
 - > <http://www.rubyonrails.org/>

Pro's And Con's

- Pro's
 - > Allows you to provide AJAX functionality within the Web application frameworks you're already using
- Con's
 - > Tied up with a particular Web application framework
- When to use
 - > If you are already using a particular Web application framework and the framework supports AJAX with an extension



Demo: Running Demos from various Web App Framework with AJAX Solutions

Demo Scenario

- Echo2
 - > <http://demo.nextapp.com/Demo/app>

So... What Should I Use?

So What Should I Use? Assuming You are using Java Tech.

- On the UI side
 - > Use AJAX-enabled JSF components whenever possible using an JSF-enabled IDE such as Sun Java Studio Creator 2
 - > If you are not ready to commit yourself to JSF component solutions yet, use jMaki
 - > If you want to have total control on the client side JavaScript coding, use Dojo toolkit
 - > If you already have Swing apps that you want to expose as AJAX-fied Web apps or if you do not want to deal with JavaScript coding, use GWT

So What Should I Use? Assuming You are using Java Tech.

- On the business logic side
 - > If you already have Java EE business logic that you want to be exposed as RMI calls on the client with AJAX behavior, use DWR
 - > If you are already using a particular Web application framework for building majority of your web application and the framework has AJAX extension, use it

AJAX Toolkits and Frameworks

Sang Shin
Java Technology Architect
Sun Microsystems, Inc.
sang.shin@sun.com
www.javapassion.com