Dojo Toolkit 1.3

Krishna Mohan Koyya
Technology Consultant & Corporate Trainer
091-9731 4231 66
krishna@glarimy.com | www.glarimy.com

Agenda

- **Breaking the Ice with Introduction & Pre-Test**
- **Session 1**
  - JavaScript Review with DOM
  - Ajax Review with JSON
- **Session 2**
  - Dojo Architecture
  - Dojo Base & Event Handling
- **Session 3**
  - Dojo Ajax
- **Session 4**
  - Dojo Drag & Drop
  - Dojo Data Abstraction
- **Session 5**
  - Dijit Overview
  - Dijit Lifecycle
- **Session 6**
  - Dijit Form Widgets
  - Dijit Layout Widgets
- **Session 7**
  - Dijit Application Widgets
- **Session 8**
  - Building & Testing
  - Performance Issues

Krishna Mohan Koyya

- **Call me Krishna**
  - Originally from Tadepalligudem in Andhra Pradesh
  - Have been in Bengaluru since 1998
- **Held various positions in the IT Industry**
  - Worked on development of large scale and complex systems
- **Extensive work in the areas of**
  - Object Orientation
  - Distributed Systems
  - Network Management Systems
  - Java Technologies

Academics

- **Graduation**
  - B.E. in Electronics & Communication Engineering
    - SRKR Engineering College, Bhimavaram, affiliated to Andhra University, Visakhapatnam
    - 1989-1993
- **Post Graduation**
  - M.Tech in Computer Science & Technology
    - College of Engineering, Andhra University, Visakhapatnam
    - 1995-1997
- **Certifications**
  - PMP (Project Management Professional)
    - PMI® USA
    - 2005
- **Others**
  - Diploma in Business Management (Finance)
    - ICFAI, Hyderabad
    - 2007

Career

- **A total of 13 years of experience with the IT Industry**
- **Since July 2008**
  - Technology Consultant & Corporate Trainer
  - Runs Glarimy, an educational services firm, based out of Bangalore, India
- **June 2007- May 2008**
  - Associate Professor & HOD of Department of IT
  - Sasi Institute of Technology and Engineering, Tadepalligudem, India
- **Jan 2006 – Dec 2006**
  - Chief Executive Officer
  - Sudhari IT Solutions India Private Limited, Bangalore, India
- **Dec 2000 – Jan 2006**
  - Software Engineer
  - Cisco Systems India Private Limited, Bangalore, India
- **Nov 1998 – Dec 2000**
  - Senior Software Engineer
  - Wipro Technologies, Bangalore, India
- **Others**
  - Dhanya Software for Hewlett-Packard ISO, Bangalore, India in 1998
  - Ace Software Exports Limited, Rajkot, India in 1997
  - Neo Software, Visakhapatnam in 1994-1995

Corporate Trainings

ORACLE

LARSEN & TOUBRO
It's all about Imagineering

hp

amazon.com

Standard Chartered
Scope International

WIPRO
Applying Thought

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

SIEMENS

Fidelity
INTERNATIONAL

First Indian Corporation
Private Limited
A Member of The First American Family of Companies

MindTree

POLARIS
live your dream

intuit.

Atos Origin
WORLDWIDE IT PARTNER

logica

verizon

SLK SOFTWARE

CDAC

icm
SINCE 1989

Training Courses

- **Basic**
  - Programming with Java 5
  - XML Technologies
  - Web development using AJAX
  - Object Oriented Analysis and Design with UML 2
  - GoF Design Patterns
- **Advanced**
  - Java Tools, Techniques and Best Practices
  - Architecting Complex Java Systems
  - JEE Web Component
  - JEE Enterprise Component using EJB 3.0
  - Axis Web Services
  - Spring Framework 2.5
  - JPA with Hibernate or OpenJPA
  - Apache Struts 2
  - Java Server Faces
  - JBoss SEAM
  - Dojo Framework
  - Agile Methodologies using XP and Scrum

About this Training

- **Schedule**
  - Morning Session: 0900-1045
  - Morning Break: 1045-1100
  - Noon Session: 1100-1245
  - Lunch Break: 1245-1345
  - After Noon Session: 1345-1530
  - Evening Break: 1530-1545
  - Evening Session: 1545-1730
- **Guidelines**
  - No permission required for leaving/entering the class room
  - Switch-off mobiles: No conversation over phone while in the class room
  - No Cross Discussions
  - Online learning resources available at www.glarimy.com
  - The training material on CD will be distributed in the last session of the training (If applicable).

Java Script: Review

- Quick Bytes: Introduction
- Code structure and Comments
- Variables
- Operators
- Programming Constructs
- Functions
- Event Handling
- User Input/Output Dialogs
- Objects
- Object: Window
- Object: Document
- Object: String
- Object: Array
- Object: Date
- Other Useful Objects
- User Defined Objects

Quick Bytes

- **A scripting language**
  - *Object Oriented*
  - *Not a full fledged programming language*
  - *No compilation required*
  - *Browsers interpret the scripts directly*
- **Brings Interactivity into web pages**
  - *Client Side Validation of User Input*
  - *Active User Interface*

Code Structure & Comments

```
<!--
<script type="text/javascript">
  //script code here
  /*
    multi line comments
  */
</script>
//-->
```

- **Code**
  - Within the html <script> tag
  - Any number of scripts in the document
  - Anywhere in the html document
  - Javascript is the default script
- **Comments**
  - <!--, //--> pair saves from old browsers from emitting the code
  - Line comments
  - Multi line comments

Variables

```
<script type="text/javascript>
  var x;
  var y = 50;
  z = 25;
  var y;
</script>
```

- **Variable Names**
  - Case sensitive
  - Should start with a letter or underscore
- **Variable Contents**
  - holds numbers, strings or objects
  - Typeof operator
    - Checks the contents of the variable
    - “number”, “string”, “boolean”, “object”, “undefined”, null
- **Variable Declaration**
  - Declaration and assignment
  - Redeclaration: No effect.
  - Auto declaration: Mere use of a variable declares itself
- **Boolean**
  - **False: <no value>, null, 0, false, “”, NaN, -0, undefined**
  - **True: <any value”, 1, true, “true”, “false”, 2**

Operators

```
<script type="text/javascript">
  x = y+z;
  s = "hi";
  s += " world";
</script>
```

- **Arithmetic Operators**
  - Addition(+), Subtraction(-), Multiplication(*), Division(/), Modulus(%), Increment(++), Decrement(--)
- **Assignment Operators**
  - =, +=, -=, *=, /=, %=
  - Concatenating strings and others with + operator
- **Comparison Operators**
  - ==, !=, >, <, >=, <=, === (equal in value and type)
- **Logical Operators**
  - && (and), || (or), ! (not)

Programming Constructs

- Branching
  - if, if … else, if … else
  - switch
- Iterations
  - for, for…in
  - while, do..while
- Flow Control
  - break
  - Continue
- Exception Handling
  - Try … catch
  - Throw

```
var x = 10;
var y = 15;
while (x < y) { x++; }
if (x == y)
    alert("Both are same");
else
    alert("They are different");
for (x=5; x<y; x++) {
    switch (x) {
        case 1: alert("One");
                break;
        case 2: alert("Two");
                break;
        default: alert("More than two");
    }
}
```

Functions

```
function area(length, breadth) {
     return (length * breadth)
}
```

- Reusable Code
- Gets executed only when ‘invoked’
- New functions can be defined
- Capable of accepting parameters
- Capable of returning values

Event Handling

```
<input type="text" name=uid size="16"
onchange="validateUserId()">
```

- Events of document elements
  - onLoad: document is loaded
  - onSubmit: documet submitted
  - onFocus: element attains focus
  - onChange: value of element changes
  - onUnload: document is unloaded
- Events and Functions
  - Event Processing is done in functions

User Interaction

```
<script type="text/javascript">
  var z=prompt("User ID")
  var a=confirm("You are " + z)
  alert(a);
  if (a=true)
        alert("Welcome" + z)
  else
        alert("Sorry! I got it wrong")
</script>
```

- **Reading User Input**
  - prompt(“message”)
  - Brings up a dialog with message and text box
  - Returns the user input to the caller
- **Confirming with the user**
  - confirm(message)
  - Returns true (for yes) and false (for no)
- **Writing Output to the User**
  - alert(message)
  - Displays the message

Objects

```
<!--
<script type="text/javascript">
  document.write(document.lastModified)
</script>
//-->
```

- Ready to use Objects
- Object Members
  - Properties
  - Methods
- Dot operator to access members
- Functions to create new objects
- Operator new to instantiate new objects

Objects: Window

- Top object in the hierarchy
- Implicit object
- Properties
  - frames[]: Collection of all named frames
  - document, history, name, parent, self, status
  - statusbar, toolbar, personalbar, scrollbar
  - closed, length, outerheigth, outerwidth
- Methods
  - alert, prompt, confirm, open, close, createPopup
  - blur, focus, print
  - resizeBy, resizeTo, moveBy, moveTo, scrollBy, scrollTo
  - setInterval, setTimout, clearInterval, clearTimeout

Objects: Document

- Properties
  - Collections
    - forms, images, links
  - lastModified: Time of last write operation
  - Referrerer: Url that referred this document
  - bgColor: Background color
  - fgColor: Foreground color (text color)
  - Title: Title of the current document
  - Domain: Domain name of the url of the current document
  - Url: URL of the current document
  - Cookie: Cookies associated with the current document
- Methods
  - write: Writes text into the html document
  - writeln: Writes a text and moves to newline
  - Stream Operations: open, close
  - getElementsByName, getElementsbyTag, getElementById

Objects: String

- **Properties**
  - length
- **Methods**
  - Display Methods:
    - Displays the string with specified presentation modifier
    - blink, bold, big, fontcolor, fontsize, italics,
    - sub, sup, toLowerCase, toUpperCase
    - link, strike
  - Manipulation Methods
    - charAt: Returns character at specified position
    - Split: Converts string into array of strings
    - Substr: Extracts specified no of chars from the string starting from 0
    - Join: Converts into string with a delimiter and returns
    - Concat: Joins two or more strings
    - indexOf: Finds first occurrence position of char in the string
    - lastIndexOf: Finds last occurrence position of char in the string
    - Replace: Replaces char with another char in the string

Objects: Array

```
var myArray = new Array();
var myOtherArray = new Array(5);
myArray[0] = "Hi";
var myYetAnotherArray = new Array("A", "BC", "DEF");
```

- **Properties**
  - length
- **Methods**
  - Concat: Joins two arrays and returns the new array
  - Join: Converts into string with a delimiter and returns
  - Sort: Sorts the elements
  - Shift: Removes and returns first element
  - Pop: Removes and returns last element
  - Push: Adds an element and returns length
  - Reverse: Reverses the array
  - toString: Converts into string and returns

Objects: Date

- **Methods**
  - Date: Returns new date with current time
  - Get methods
    - getDate, getDay, getMonth, getYear, getFullYear,
    - getHours, getMinutes, getSeconds, getTime, getMilliSeconds, getTimezoneOffset
    - getUTCDate, getUTCDay, getUTCFullYear, getUTCMonth,
    - getUTCHours, getUTCMinutes, getUTCSeconds, getUTCMilliSeconds
  - SetMethods
    - Set methods for corresponding get methods
  - Parse: Returns time in milliseconds since 1-1-1970
  - UTC: Same as parse according to universal time

Other Useful Objects

```
var browser=navigator.appName;
var b_version=navigator.appVersion;
var version=parseFloat(b_version);
document.cookie= "nameCookie=krishna";
If (document.cookie.indexOf("nameCookie=") != -1)
  //retrieve it
```

- **Navigator**
  - Access to browser info
  - Methods: appName, appVersion
- **Cookies**
  - Name cookie, Password cookie, Date cookie
  - Read and write on document.cookie
  - Document.cookie is a collection

User Defined Objects

```
function MyClass(name, city) {
        this.name = name;
        this.city = city;
}
myObject = new MyClass("Krishna", "Bangalore");
function print() {
        //do something
}
myOtherObject = new Object();
myOtherObject.title = "JavaScript";
myOtherObject.author = "Myself"
myOtherObject.print = print;
```

- **Creation of object directly**
  - objectName = new Object()
  - objectName.propertyName = value
  - objectName.methodName = methodName
- **Creation of object using template**
  - function objectTemplateName(properties)
  - Use this operator in the function
  - new operator to create instances

Ajax: Review

- What is AJAX?
- A Simple Example
  - The Old Way
  - The AJAX Way
- Key Aspects
- Advantages of the AJAX Approach?
- AJAX Technologies
- The Basic AJAX API
- Creating the XMLHttpRequest Object
- The XMLHttpRequest Object Basics
- Complete Example

Ajax: Quick Bytes

- AJAX
  - Asynchronous JavaScript and XML.
  - A new technique for creating better, faster, and more interactive web applications
  - Uses XHTML for content and CSS for presentation
  - Uses Document Object Model and JavaScript for dynamic content display.
  - A web browser technology independent of web server software.
  - Intuitive and natural user interaction.
  - Data-driven as opposed to page-driven

- Conventional web application
  - Transmits information to and from the sever using synchronous requests.

- With AJAX
  - When submit is pressed, JavaScript will make a request to the server, interpret the results and update the current screen.
  - A user can continue to use the application while the client program requests information from the server in the background
  - No clicking required only Mouse movement is a sufficient event trigger.

Ajax Technologies

- **JavaScript**
  - Loosely typed scripting language
  - JavaScript function is called when an event in a page occurs
  - Glue for the whole AJAX operation
- **DOM**
  - API for accessing and manipulating structured documents
  - Represents the structure of XML and HTML documents
- **CSS**
  - Allows for a clear separation of the presentation style from the content and may be changed programmatically by JavaScript
- **XMLHttpRequest**
  - JavaScript object that performs asynchrous interaction with the server

Ajax Way

- A client event occurs
- An XMLHttpRequest object is created
- The XMLHttpRequest object is configured
- The XMLHttpRequest object makes an asynchronous request to the Webserver.
- Webserver returns the result containing XML document.
- The XMLHttpRequest object calls the callback() function and processes the result.
- The HTML DOM is updated

Ajax Impact

- **Increased complexity**
  - Server side developers will need to understand that presentation logic will be required in the HTML client pages as well as in the server-side logic
  - Page developers must have JavaScript technology skills
- **Difficult to debug, test, and maintain**
  - JavaScript is hard to test - automatic testing is hard
  - Weak modularity in JavaScript
  - Lack of design patterns or best practice guidelines yet
- **No standardization of the XMLHttpRequest yet**
  - Future version of IE will address this
- **No support of XMLHttpRequest in old browsers**
  - Iframe will help
- **JavaScript technology dependency & incompatibility**
  - Must be enabled for applications to function
  - Still some browser incompatibilities
- **JavaScript code is visible to a hacker**
  - Poorly designed JavaScript code can invite security problem

XHR Interface

```
interface XMLHttpRequest {
        readonly attribute unsigned short readyState;
        attribute DOMString responseText;
        attribute Document responseXML;
        attribute unsigned short status;
        attribute DOMString statusText;

        void open(in DOMString method, in DOMString uri);
        void open(in DOMString method, in DOMString uri, in boolean async);
        void open(in DOMString method, in DOMString uri, in boolean async, in
            DOMString user);
        void open(in DOMString method, in DOMString uri, in boolean async, in
            DOMString user, in DOMString password);
        void setRequestHeader(in DOMString header, in DOMString value)
            raises(DOMException);
        void send(in DOMString data) raises(DOMException);
        void send(in Document data) raises(DOMException);
        void abort();
        DOMString getAllResponseHeaders();
        DOMString getResponseHeader(in DOMString header);
        attribute Function onreadystatechange;
};
```

XHR: Attributes

- **onreadystatechange of type Function**
  - To be invoked when readyState changes value. To be invoked multiple times when readyState is 3.
- **readyState of type unsigned short, readonly**
  - 0 Uninitialized: The initial value.
  - 1 Open: The open() method has been successfully called.
  - 2 Sent: The UA successfully completed the request, but no data has yet been received.
  - 3 Receiving: Immediately before receiving the message body (if any). All HTTP headers have been received.
  - 4 Loaded: The data transfer has been completed.
- **status of type unsigned short**
  - HTTP Status Code for Receiving and Loaded
  - Raises exception DOMExceptionINVALID_STATE_ERR for other states
- **statusText of type DOMString**
  - HTTP Status Text for Receiving and Loaded
  - Raises exception DOMExceptionINVALID_STATE_ERR for other states

XHR Attributes

- **responseText of type DOMString**
  - Body of the data received so far for states Receiving and Loaded.
  - Empty string for other states.
  - Character encoding: As specified or UTF-8.
- **responseXML of type Document**
  - Null, for all states other than Loaded.
  - Object of Document interface representing the parsed document, for text/xml, application/xml documents.
  - Null, if not an XML document or the XML is not well-formed.

XHR Methods

- **abort**
  - Cancels network activity of the object
  - Resets the object
  - No Parameters
  - No Return Value
  - No Exceptions
- **getAllResponseHeaders**
  - For Receiving and Loaded
    - Returns all CRLF separated HTTP headers in single string
  - Returns null for others
- **getResponseHeader**
  - For Receiving and Loaded
    - Returns COMMA SPACE separated values for the selected HTTP Header.
  - Empty string for others.
- **setRequestHeader**
  - Throw exception for not Open
  - Ignore headers/values with LF, CR, SPACE and COLON
  - Sets/concatenates request header header to value (COMMA,SPACE)

XHR Methods

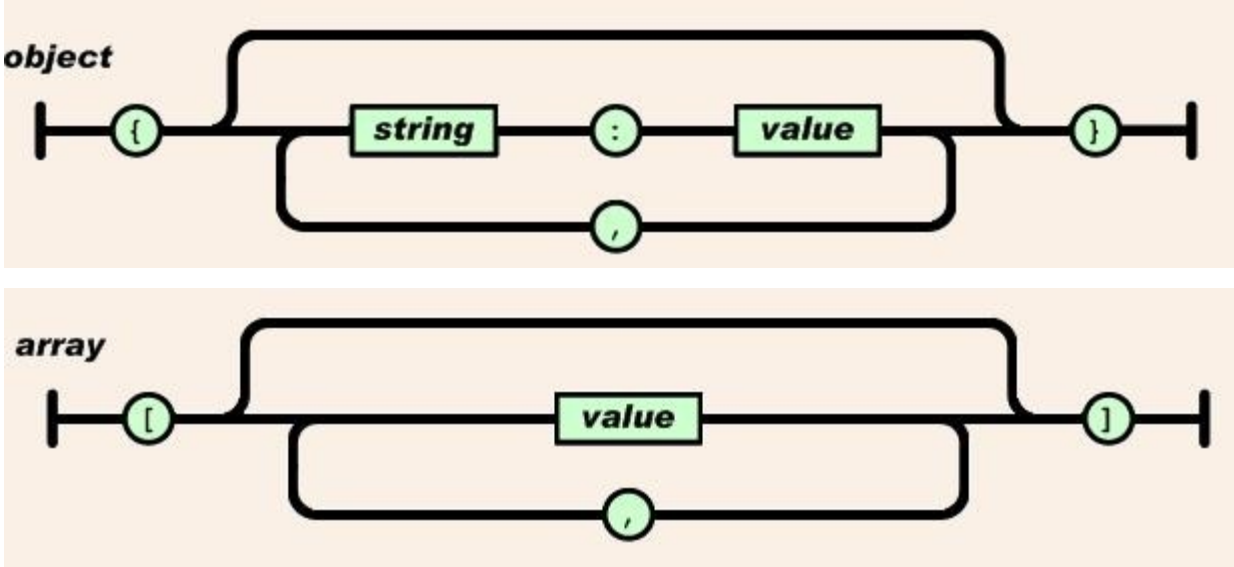
- **open**
  - Sets readyState to Open
  - Resets responseText, responeXML, status, statusText
  - Applies same-origin security restrictions
  - For Loaded state, reset everyting.
- **Send**
  - For Open: throw exception
  - For others:
    - Sets readyState to Sent
    - Sends request to the URI
    - Returns immediately if async is true
    - Waits if async is false
  - For string data: use UTF-8
  - For Document data: serialize it using data.xmlEncoding or UTF-8
  - Uses error page as the response for errors.
  - Sets to Receiving before processing message body
  - Sets to Loaded after completion

JSON: Quick Bytes

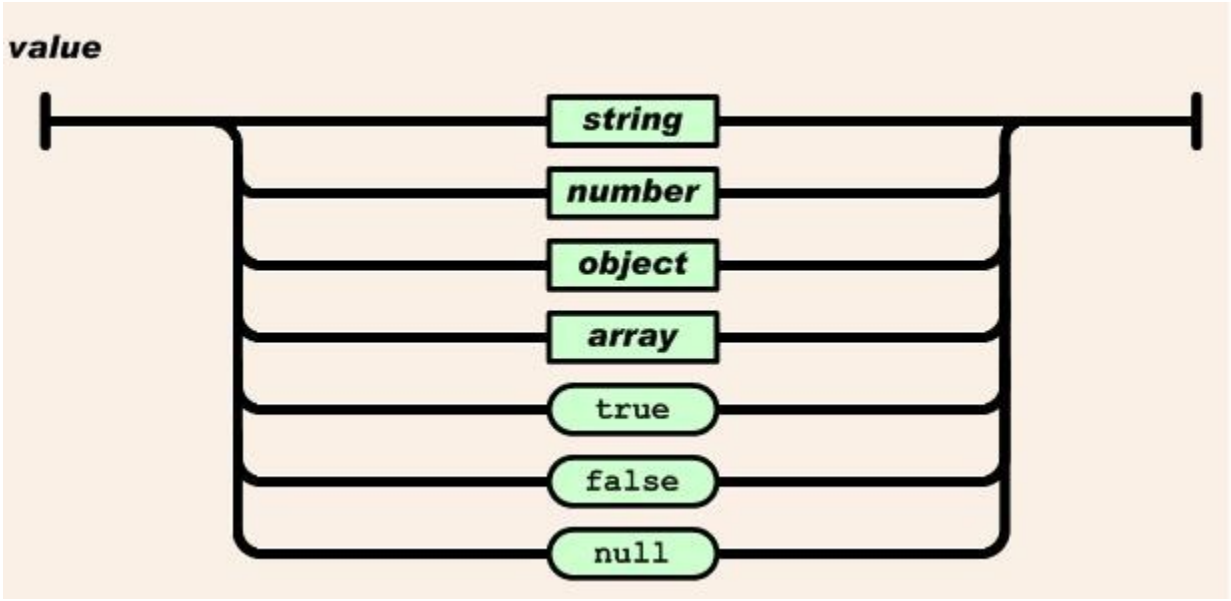
- **JavaScript Object Notation**
  - lightweight data-interchange format
  - easy for humans to read and write
  - easy for machines to parse and generate
  - based on a subset of the JavaScript
  - is a text format
  - is completely language independent
- Built on two structures
  - A collection of name/value pairs. In various languages, this is realized as an *object*, record, struct, dictionary, hash table, keyed list, or associative array.
  - An ordered list of values. In most languages, this is realized as an *array*, vector, list, or sequence.

JSON Arrays & Objects

- **Built on two structures**
  - A collection of name/value pairs. In various languages, this is realized as an *object*, record, struct, dictionary, hash table, keyed list, or associative array.
  - An ordered list of values. In most languages, this is realized as an *array*, vector, list, or sequence.



JSON Value



JSON Examples

- **Array**
  - var Beatles = ["Paul","John","George","Ringo"];
  - var Beatles = new Array("Paul","John","George","Ringo");
- **Object**
  - var Beatles = { "Country" : "England", "YearFormed" : 1959, "Style" : "Rock'n'Roll" }
  - var Beatles = new Object(); Beatles.Country = "England"; Beatles.YearFormed = 1959; Beatles.Style = "Rock'n'Roll";
  - alert(Beatles.Style); //Dot Notation
    alert(Beatles["Style"]); //Bracket Notation

JSON Examples

- **Arrays in Object**
  - var Beatles = { "Country" : "England", "YearFormed" : 1959, "Style" : "Rock'n'Roll", *"Members" : ["Paul","John","George","Ringo"]* }

- **Objects in Array**
  - var Rockbands = [*{ "Name" : "Beatles", "Country" : "England", "YearFormed" : 1959, "Style" : "Rock'n'Roll", "Members" : ["Paul","John","George","Ringo"] }, { "Name" : "Rolling Stones", "Country" : "England", "YearFormed" : 1962, "Style" : "Rock'n'Roll", "Members" : ["Mick","Keith","Charlie","Bill"] }*]

JSON Under Scanner

- **Advantages**
  - **JavaScript Syntax**
    - The fact that JSON uses JavaScript syntax is the biggest advantage. It makes it extremely easy to work with on the client side. Compare this to traversing the XML DOM or XSLT and you'll find great efficiencies in working with JSON.
  - **Brevity**
    - Another oft-touted advantage of JSON is that it is lightweight. Compare, for example, the Rockbands object we saw earlier in the lesson in JSON vs. XML:
- **Disadvantages**
  - **New Syntax - Not So Human Readable**
    - The biggest disadvantage of working with JSON is it is yet another syntax we need to familiarize ourselves with. However, with a little effort, most developers will find it's not too difficult to get comfortable with.
- **Not "XPathable"**
    - XPath makes it possible to search through XML documents and find specific node values based on many different criteria (e.g, relative location to other nodes in the document). JavaScript objects and arrays have no comparable built-in capability.

Java & JSON

Source	Javadoc	Description
JSONObject.java	JSONObject.html	A JSONObject is an unordered collection of name/value pairs. Its external form is a string wrapped in curly braces with colons between the names and values, and commas between the values and names. The internal form is an object having get() and opt() methods for accessing the values by name, and put() methods for adding or replacing values by name. The values can be any of these types: Boolean, JSONArray, JSONObject, Number, and String, or the JSONObject.NULL object.
JSONArray.java	JSONArray.html	A JSONArray is an ordered sequence of values. Its external form is a string wrapped in square brackets with commas between the values. The internal form is an object having get() and opt() methods for accessing the values by index, and put() methods for adding or replacing values. The values can be any of these types: Boolean, JSONArray, JSONObject, Number, and String, or the JSONObject.NULL object.
JSONStringer.java	JSONStringer.html	A JSONStringer is a tool for rapidly producing JSON text.
JSONWriter.java	JSONWriter.html	A JSONWriter is a tool for rapidly writing JSON text to streams.
JSONTokener.java	JSONTokener.html	A JSONTokener takes a source string and extracts characters and tokens from it. It is used by the JSONObject and JSONArray constructors to parse JSON source strings.
JSONException.java	JSONException.html	A JSONException is thrown when a syntax or procedural error is detected.
JSONString.java	JSONString.html	The JSONString is an interface that allows classes to implement their JSON serialization.

DoJo ToolKit

Dojo Toolkit

- Dojo Architecture
- Dojo Base
- Dojo Language Utilities
- Dojo Eventing
- Dojo Ajax
- Dojo DnD
- Dojo Data Abstraction
- Dijit Architecture
- Dijit Layouts & Widgets
- Other Dojo Stuff

Dojo Architecture

- **Base, the kernel**
  - dojo/dojo.js
  - Requires always
  - JavaScript Library
- **Core, the script library**
  - dojo
  - Requires More Often
  - Animation, Data, Internationalization and etc.
- **Dijit, the widget library**
  - dijit
  - Application and Form widjet library
- **DojoX, to top it all**
  - dojox
  - extenstions and experiments

Dojo Environment

- **Dojo 1.3.1, the runtime**
  - http://download.dojotoolkit.org
  - Subversion
  - http://o.aolcdn.com/dojo/1.3.1/dojo/dojo.xd.js
- **Firefox 3, the browser**
  - http://www.getfirefox.com
- **Firebug, the debugger**
  - http://www.getfirebug.com
- **Tomcat 5.0, the web server**
  - http://tomcat.apache.org
- **EditPlus, the editor**
  - http://www.editplus.com/download.html
- **Everything is already provided to you**
  - **Except Firebug**

Language Utilities

- **Node Selection**
  - Dojo.byId
- **Type Checking**
  - Dojo.isString
  - Dojo.isArray
  - Dojo.isFunction
  - Dojo.isObject
- **String Utilities**
  - Dojo.string.trim
  - Dojo.string.substitute
  - Dojo.string.pad
- **Array Utilities**
  - Dojo.indexOf
  - Dojo.lastIndexOf
  - Dojo.every
  - Dojo.some
  - Dojo.map
  - Dojo.filter

Other Utilities

- **Resource Mapping**
  - Dojo.provide
  - Dojo.require
  - Dojo.registerModulePath
- **Object Utilities**
  - Dojo.extend(newObject, object1, object2)
  - Dojo.clone(object)
- **DOM Utilities**
  - Dojo.isDescendant(string/node, string, node)
  - Dojo.setSelectable(boolean)
  - Dojo.style(name, {style definition})
  - Dojo.attr(node, attr)
  - Dojo.attr(node, attr, value)
  - Dojo.hasAttr(node, attr)
  - Dojo.removeAttr(node, attr)
  - Dojo.place(newNode, existingNode, position)
- **Browser Utilities**
  - Dojo.isMozilla
  - Dojo.isIE

Event Handling

- **Events**
  - Onclick
  - Onmousedown
  - Onmouseup
  - Onmouseover
  - Onmouseout
  - Onmousemove
  - Onkeydown
  - Onkeyup
  - Onkeypress
- **Synchronous Events**
  - Dojo.connect(sourceObject, sourceFunction, reactObject, reactFunction)
  - Dojo.disconnect(handle)
- **Asynchronous Events**
  - Dojo.publish(topic, args)
  - Dojo.subscribe(topic, function)
  - Dojo.unsubscribe(handle)

XHR and Ajax

- **xhrGet**
  - url: url
  - load: callback
  - error: on error
  - content: json
- **xhrPost**
  - url: url
  - load: callback
  - error: on error
  - form: form name
- **Deferred**
  - addCallback(function)
  - addErrback(function)
  - addBoth(function)
  - addCallbacks(cbFunction, ebFunction)
  - callback
  - erroback
  - cancel

Drag and Drop

- **Terminology**
  - Container
  - Nodes
  - Selector
  - Mover
  - Movable
  - Source
  - Target
  - Avatar
  - DnD Manager
- **Dojo Support**
  - Dojo.dnd.Source
    - dndType
  - Dojo.dnd.Target
    - accept
  - Dojo.dnd.Moveable

Drag and Drop Actions

- **Publish/Subscribe**
- **Topics**
  - /dnd/source/over
  - /dnd/start
  - /dnd/stop
  - /dnd/cancel
- **Parameters**
  - Source
    - Source of the drag and drop
  - Nodes
    - Nodes being dragged

Data Abstraction

- **Dojo.data**
  - Abstraction over data sources
  - May involve Ajax
- **Comparison**
  - Data Store Vs ResultSet
  - Data Source Vs Database
  - Item Vs Row
  - Attribute Vs Column
  - Identity Vs Primary Key
  - Dojo.data API Vs JDBC
  - Query Vs WHERE
  - Request Vs SELECT

Data Abstraction API

- **Dojo.data.api.Read**
  - **Supports read, search, sort and filter**
- **Dojo.data.api.Write**
  - **Supports create, delete, update**
- **Dojo.data.api.Identity**
  - **Supports primary key**
- **Dojo.data.api.Notification**
  - Notifies event listeners for create, delete & update
- **Store Functions**
  - Store.getValue(item, attribute)
  - store.getFeatures
  - Store.getAttributes(item)
  - Store.fetch()
  - Store.save()
  - Store.deleteItem()

Available Stores

- **ItemFileReadStore**
  - JSON Read Only
- **ItemFileWriteStore**
  - JSON Write Only
- **CsvStore**
  - CSV Read Only
- **HtmlTableStore**
  - Table Data Read Only
- **XmlStore**
  - XML Read/Write
- **FlickrStore**
  - Flicker Webservices Read Only

ItemFileReadStore

- **API**
  - **Dojo.data.api.Read**
  - **Dojo.data.api.Identity**
- **Reads JSON**
  - From HTTP Endpoints
  - From in-memory Javascript objects
- **Stores content in memory**
- **Constructor**
  - url, data and typeMap
- **Structure**
  - Identifier, label, items

ItemFileWriteStore

- **API**
  - **Dojo.data.api.Read**
  - **Dojo.data.api.Identity**
  - **Dojo.data.api.Write**
  - **Dojo.data.api.Notification**
- **Reads/Write JSON**
  - From/To HTTP Endpoints
  - From/To in-memory Javascript objects
- **Stores content in memory**
- **Constructor**
  - url, data and typeMap
- **Structure**
  - Identifier, label, items

Dijit

- **Widget Library**
- **Parser**
  - djConfig=parseOnLoad:true
  - dojo.parser.parse();
- Parsing
  - Queries for dojo type
  - Fetches class information
  - Detects dojo/connect & dojo/method
  - Creates class instance
  - Maps jsId as a global varibale
  - Processes dojo/connect & dojo/method

Dijit: Form Widgets

- **Form Dijits**
  - Form
  - Button variations
  - ComboBox
  - FilteringSelect
    - Drop-in SELECT element
  - NumberSpinner
    - Increment/decrement numbers
  - Slider
  - TextArea
    - Resizable
  - SimpleTextArea
  - MultiSelect
  - TextBox variations

Dijit: Layout Widgets

- **Layout Widgets**
  - Content Pane
    - Sits in container
  - Tab Container
    - Shows one tab at a time
  - Stack Container
    - Shows one slide at a time
  - Accordion Container
    - Shows one tile at a time
  - Border Container / Layout Container
    - Top, bottom, left, right, client

Dijit: Application Widgets

- **Application Widgets**
  - Menu
  - Toolbar
  - Dialog
  - TooltipDialog
  - ProgressBar
  - TitleBar
  - Tooltip
  - InlineEditBox
  - ColorPalette
  - Editor
  - Tree

Dijit: Lifecycle Methods

- **preamble**
  - To manipulate args to constructor
- **constructor**
  - To override object construction
- **postMixinProperties**
  - After inheriting properties from the hierarchy
- **buildRendering**
  - To override connecting the object with DOM node and placing on the page
- **postCreate**
  - After rendering the widget
- **startup**
  - After all child widgets are created
- **destroyRecursive**
  - Calls uninitialize
- **uninitialize**
  - To override teardown

Dijit: Progress Bar

- Indeterminate
  - Boolean
  - Progress display
- Maximum
  - Number of points
  - Default 100
- Progress
  - % or absolute values
- Update(progress)
  - Progress information

Dijit: Dialog

- Plain Dialogs
- Tooltip Dialogs
- Show()
- Hide()
- Execute
- Fields

Dijit: Tree

- Tree & Forest
- Events
- Query
- Store
  - Children

Dijit: Editor

- Focus On Load
- Height
- Minimum Width
- Inherit Width
- Minimum Height
- Name
- GetValue
- Replacevalue
- Close

Dojo Builds

- **Build process**
  - **Layering**
    - Grouping modules together
  - **Interning non-java-script files**
    - Assigning them to strings
  - **Compressing layers down**
    - White spaces, variable names and etc
- **Profile**
- **Pre-requisites**
  - Java
  - Dojo Source Build on Subversion

Dojo Build Profile

```
dependencies ={
        layers:  [
            {
            name: "mydojo.js",
            dependencies: [
                "dijit.Button",
                "dojox.wire.Wire",
                "dojox.wire.XmlWire",
                "explosive.space.Modulator"
            ]
            }
        ],
        prefixes: [
            [ "dijit", "../dijit" ],
            [ "dojox", "../dojox" ],
            [ "explosive", "../../explosive" ]
        ]
    };
```

Dojo Building

- **Building**
  - Util/buildscripts/build.bat [options]
- **Options**
  - Profile
  - Profile File
  - Action (clean/release)
  - Version: 0.0.0.dev
  - Release Name: dojo
  - Release Directory
  - Loader: xdomain
  - Optimize: comments/shrinksafe
  - copyTests: true
  - xdDojoPath

Performance Optimization

- **Use Dojo custom builds**
  - Reduces latency
- **Lazy Load**
  - Don’t load entire stuff in a go
- **Use static pages**
  - Let Ajax fill up the dynamic portions
- **Cache Javascript files**
  - Have maximum retention
- **Use Data Compression**
  - If possible
- **Reduce Tags**
  - Use CSS

After Training Service

- **Weekly News Letter**
  - To all the participants
  - Update on what's new on the portal
- **Phone**
  - 091-9731 4231 66
  - For appointments
- **E-mail**
  - krishna@glarimy.com
  - For any technical queries
- **Portal**
  - http://www.glarimy.com
  - FAQ, Tutorials, Presentations, Samples, Quiz

Thank You