Introduction to Dojo Toolkit

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
- Many slides are created from the contents posted in dojotoolkit.org website

Topics

- What is and Why Dojo Toolkit?
- Dojo Toolkit Package System
- Remoting via dojo.io.bind
- Dojo DOM Manipulation
- Backward/Forward buttons, Bookmarking
- Dojo Event System
  - Overview
  - DOM events
  - Chaining function calls
  - AOP event model
- Usage of Dojo widgets

Topics Covered in Advanced Dojo Presentation

- Creation of Dojo Widgets
- Dojo Drag and Drop
- Dojo Animation
- Dojo Storage
- Performance tuning

What is and Why Dojo Toolkit?

What is Dojo Toolkit?

- Open Source DHTML toolkit written in JavaScript
  - It is a set of JavaScript libraries
- Aims to solve some long-standing historical problems with DHTML
  - Browser incompatibility
- Allows you to easily build dynamic capabilities into web pages
  - Widgets
  - Animations
- Server technology agnostic

source: dojotoolkit.org

Why Dojo Toolkit?

- You can use Dojo to make your web applications more useable, responsive, and functional
  - Supports AJAX
- Dojo provides lots of plumbing facilities
  - Hides XMLHttpRequest processing
  - Handles browser incompatibilities
- Strong developer community

source: dojotoolkit.org

Features of Dojo Toolkit

- Powerful AJAX-based I/O abstraction (remoting)
- Graceful degradation
- Backward, forward, bookmarking support
- Aspect-oriented event system
- Markup-based UI construction through widgets
- Widget prototyping
- Animation
- Lots of useful libraries

Dojo Toolkit Package System

Dojo Toolkit Package System

- Dojo libraries are organized in packages just like Java libraries are
- You import only the packages you need
  - dojo.require("dojo.event.*");
  - dojo.require("dojo.dnd.*");
- The require() will go out and dynamically retrieve the JavaScript code and load them up in the page
- You can write your own packages

Dojo Toolkit Libraries

- dojo.io: AJAX-based communication with the server
- dojo.event: unified event system for DOM and programmatic events
- dojo.lang: utility routines to make JavaScript easier to use.
- dojo.string: String manipulation routines
- dojo.dom: DOM manipulation routines
- dojo.style: CSS Style manipulation routines

Dojo Toolkit Libraries

- dojo.html: HTML specific operations
- dojo.reflect: Reflection API
- dojo.date: Date manipulation
- dojo.logging.Logger: Logging library
- dojo.profile: JS Profiler
- dojo.regexp: Regular Expression generators
- dojo.dad: Drag and Drop

Dojo Toolkit Libraries

- dojo.collections: Collection data structures
- dojo.crypto: Cryptographic API
- dojo.reflection: Reflection routines
- dojo.math: Mathematic routines
- dojo.storage: Storage routines
- dojo.uri: URL handling routines
- dojo.widget: Widget framework

dojo.io.bind()

dojo.io.bind()

- The dojo.io.* namespace contains generic APIs for doing network I/O
  - dojo.io.bind() hides low-level XMLHttpRequest operations
  - In a browser, dojo.io.bind() is how one makes "Ajax" requests
- Also handles
  - back-button interception
  - transparent form submission
  - advanced error handling

dojo.io.bind() Syntax

```
// Make a request that returns raw text from a URL
dojo.io.bind({
    // URL to make a request to
    url: "http://foo.bar.com/something",

    // Callback function to execute upon successful response
    load: function(type, data, evt){ /*do something w/ the data */ },

    // Type of data that is returned
    mimetype: "text/plain"

    // More options
});
```

source: dojotoolkit.org

Example #1: Simple Request

```
// Make a request that returns raw text from a URL.
dojo.io.bind({
    url: "http://foo.bar.com/sampleData.txt",
    load: function(type, data, evt){ /*do something w/ the data */ },
    mimetype: "text/plain"
});
```

source: dojotoolkit.org

Example #2: Simple Request

```
// Create an argument first
var bindArgs = {
    url: "http://foo.bar.com/sampleData.txt",
    load: function(type, data, evt){ /*do something w/ the data */ },
    mimetype: "text/plain"
};

// dispatch the request
dojo.io.bind(bindArgs);
```

source: dojotoolkit.org

Example #2: Error Handling

```
// Make a request that returns raw text from a URL
// with error handling
dojo.io.bind({
    url: "http://foo.bar.com/sampleData.txt",
    load: function(type, data, evt){ /*do something w/ the data */ },
    error: function(type, error){ /*do something w/ the error*/ },
    mimetype: "text/plain"
});
```

source: dojotoolkit.org

Example #3: Same as Example #2

```
// Handle error condition using type
dojo.io.bind({
    url: "http://foo.bar.com/sampleData.txt",
    handle: function(type, data, evt){
        if(type == "load"){
            // do something with the data object
        }else if(type == "error"){
            // here, "data" is our error object
            // respond to the error here
        }else{
            // other types of events might get passed, handle them here
        }
    },
    mimetype: "text/plain"
});
```

source: dojotoolkit.org

mimetype - Specifies the response format

- text/plain
  - response is in string format
- text/javascript
  - response is in JavaScript format
- text/xml
  - response is in XML format
- text/json
  - response is in JSON format

Example #4: Dynamic Content Loading (Getting JavaScript string)

```
// Request a JavaScript literal string and then evaluate it.
// That's also baked into bind, just provide a different expected
//  response type with the mimetype argument:
dojo.io.bind({
    url: "http://foo.bar.com/sampleData.js",
    load: function(type, evaldObj){ /* do something */ },
    mimetype: "text/javascript"
});
```

source: dojotoolkit.org

Example #5: Explicitly Specify XMLHTTP Transport

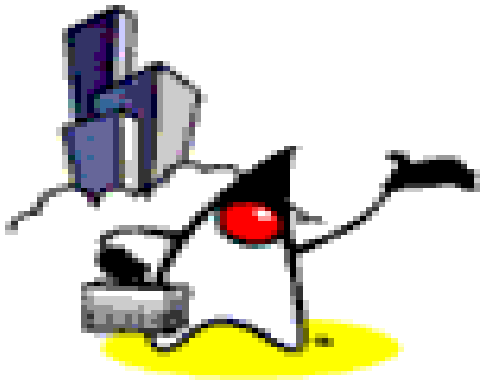
```
// If you want to be DARN SURE you're using the XMLHTTP
// transport, specify XMLHTTP Transport.
dojo.io.bind({
    url: "http://foo.bar.com/sampleData.js",
    load: function(type, evaldObj){ /* do something */ },
    mimetype: "text/plain", // get plain text, don't eval()
    transport: "XMLHTTPTransport"
});
```

source: dojotoolkit.org

Example #6: Submission of forms via a request

```
// Being a jack-of-all-trades, bind() also supports the submission
// of forms via a request (with the single caveat that it won't do
// file upload over XMLHTTP):
dojo.io.bind({
    url: "http://foo.bar.com/processForm.cgi",
    load: function(type, evaldObj){ /* do something */ },
    formNode: document.getElementById("formToSubmit")
});
```

source: dojotoolkit.org



Demo: Remoting with dojo.io.bind()

dojo.io.bind():
Back/Forward Buttons & Bookmarking

Example #6: Back Button

```
var sampleFormNode = document.getElementById("formToSubmit");

dojo.io.bind({
    url: "http://foo.bar.com/sampleData.js",
    load: function(type, evaldObj){
        // hide the form when we hear back that it submitted successfully
        sampleFormNode.style.display = "none";
    },
    backButton: function(){
        // ...and then when the user hits "back", re-show the form
        sampleFormNode.style.display = "";
    },
    formNode: sampleFormNode
});
```

source: dojotoolkit.org

Example #7: Forward Button

```
var sampleFormNode = document.getElementById("formToSubmit");

dojo.io.bind({
    url: "http://foo.bar.com/sampleData.js",
    load: function(type, evaldObj){
        // hide the form when we hear back that it submitted successfully
        sampleFormNode.style.display = "none";
    },
    backButton: function(){
        // ...and then when the user hits "back", re-show the form
        sampleFormNode.style.display = "";
    },
    forwardButton: function(){
        // and if they hit "forward" before making another request, this
        // happens:
        sampleFormNode.style.display = "none"; // we don't re-submit
    },
    formNode: sampleFormNode
});
```

source: dojotoolkit.org

Example #8: Bookmarking

- Simply pass in the value "true" to the flag "changeURL"
- Your top-level page will now have a new timestamp hash value appended to the end

```
dojo.io.bind({
    url: "http://foo.bar.com/sampleData.txt",
    load: function(type, data, evt){ /*do something w/ the data */ },
    changeURL: true,
    mimetype: "text/plain"
});
```

source: dojotoolkit.org

Example #9: Bookmarking

- Alternately, it's possible to include your own hash value by providing a string instead of "true" to changeURL
- http://foo.bar.com/howdy.php#foo,bar,baz

```
dojo.io.bind({
    url: "http://foo.bar.com/sampleData.txt",
    load: function(type, data, evt){ /*do something w/ the data */ },
    changeURL: "foo,bar,baz",
    mimetype: "text/plain"
});
```

source: dojotoolkit.org

Dojo Event System:
Transport

Transports

- dojo.io.bind and related functions can communicate with the server using various methods, called transports
  - XMLHttp
  - IFrame I/O
  - ScriptSrcIO
- Each has certain limitations, so you should pick the transport that works correctly for your situation
- The default transport is XMLHttp

source: dojotoolkit.org

XMLHttp Transport

- It works well in most cases, but it cannot transfer files, cannot work across domains (ie, cannot connect to another site than the current page), and doesn't work with the file:// protocol

```
dojo.require("dojo.io.*");

function mySubmit() {
  dojo.io.bind ({
    url: 'server.cfm',
    handler: callBack,
    formNode: dojo.byId('myForm')
  });
}

function callBack(type, data, evt) {
   dojo.byId('result').innerHTML = data;
}
```

source: dojotoolkit.org

IFrame I/O Transport

- The IFrame I/O transport is useful because it can upload files to the server

```
dojo.require("dojo.io.*");
dojo.require("dojo.io.IframeIO");

function mySubmit() {
  dojo.io.bind ({
    url: 'server.cfm',
    transport: “IframeTransport”,
    handler: callBack,
    formNode: dojo.byId('myForm')
  });
}

function callBack(type, data, evt) {
   dojo.byId('result').innerHTML = data;
}
```

source: dojotoolkit.org

ScriptSrcIO Transport

- Due to security restrictions, XMLHttp cannot load data from another domain
- The ScriptSrcIO? transport is useful for doing this
- Yahoo's RPC service is implemented using ScriptSrcIO?

```
dojo.require("dojo.io.*");
dojo.require("dojo.io.ScriptSrcIO");

dojo.io.bind({
    url: "http://example.com/json.php",
    transport: "ScriptSrcTransport",
    jsonParamName: "callback",
    mimetype: "text/json",
    content: { ... }
});
```

source: dojotoolkit.org

DOM Manipulation

Dojo DOM Manipulation Routines

- dojo.byId(“someid”)
  - Same as document.getElementById("someid");
- dojo.dom.isNode(node)
- dojo.dom.getTagName(node)
- dojo.dom.firstElement(node)
- dojo.dom.lastElement(node)
- dojo.dom.nextElement(node)
- dojo.dom.prevElement(node)

source: j2ee blueprints

Dojo DOM Manipulation Routines

- ddojo.dom.moveChildren (srcNode, destNode, trim)
- dojo.dom.copyChildren (srcNode, destNode, trim)
- dojo.dom.removeChildren(node)
- dojo.dom.replaceChildren(node, newChild)
- dojo.dom.removeNode(node)
- dojo.dom.getAncestors(node, filterFunction, returnFirstHit)
- dojo.dom.getAncestorsByTag(node, tag)
- dojo.dom.innerXML(node)

source: j2ee blueprints

Dojo DOM Manipulation Routines

- dojo.dom.createDocumentFromText(str, mimetype):
- dojo.dom.insertAfter(node, reference, force):
- dojo.dom.insertAtPosition(node, reference, position)
- dojo.dom.textContent(node)
  - Gets the text-only serialization of a node's children
- dojo.dom.textContent(node, text)
  - Sets the text-only serialization of a node's children

source: j2ee blueprints

Dojo Event System

Dojo Event System Topics

- Overview
- Handling DOM events
- Chaining function calls
- Before advice (Aspected-oriented event system)
- Disconnecting event handlers
- Resources

source: j2ee blueprints

Dojo Event System:
Overview

JavaScript Event Handling

- Events are essential in JavaScript components because
  - as they drive the user interface
  - result in AJAX requests
  - allow JavaScript components to interact with each other

source: j2ee blueprints

Issues of JavaScript Event Handling

- Cross browser event handling code is difficult to write from scratch
  - There are various ways in JavaScript of handling events and each browser has its own quirks and issues
- As the number JavaScript components in a page increases, the component code can tend to become more tightly coupled, thus less maintainable
- Attaching multiple event handlers to a node is hard
  - The previously attached event handler is overwritten by a new one

Why Dojo Event System?

- It abstracts the JavaScript event system
  - Lets you register to "hear" about any action through a uniform and simple to use API - dojo.event.connect()
- Treat any function call as an event that can be listened to
  - Handles more than simple DOM events
- It provides advanced event handling schemes
  - Aspect oriented - your event handler can be called “before” or “after”
- Less unobtrusive
  - The setting of event handlers on DOM Nodes happen without explicit on* attributes in markup

dojo.event.connect(srcObj, "srcFunc", "targetFunc")

- Provides a uniform API for event handling
- Abstracts browser differences
- Prevents memory leaks that appear on some browsers
- Takes care of the details of attaching more than one event handler (multiple event handlers) to a single event type

Dojo Event System:
Handling DOM Events

Example #1 - DOM event Using Named Event Handler

```
<script type="text/javascript" src="dojo.js"></script>
<script type="text/javascript">
window.onload = function () {
  var link = document.getElementById("mylink");

  // “myHandler” event handler is registered for the
  // "onclick" property of “mylink” node.
  dojo.event.connect(link, "onclick", myHandler);
}

// Define an event handler named as “myHandler”
function myHandler(evt) {
    alert("myHandler: invoked - this is my named event handler");
}
</script>
<a href="#" id="mylink">Click Me</a>
```

source: j2ee blueprints

Example #2 Using Unnamed (Anonymous) Event Handler

```
<script type="text/javascript" src="dojo.js"></script>
<script type="text/javascript">
window.onload = function () {
   var link = document.getElementById("mylink");
   // connect link element 'onclick' property to an anonymous function
   dojo.event.connect(link, "onclick", function(evt) {
      var srcElement;
      if (evt.target) {
          srcElement = evt.target;
      } else if (evt.srcElement) {
          srcElement = evt.srcElement;
      }
      if (srcElement) {
         alert("dojo.event.connect event: " + srcElement.getAttribute("id"));
      }
   });
 }"
</script>
<a href="#" id="mylink" onclick="alert('inline event');">Click Me</a>
```

source: j2ee blueprints

Example #3: Attaching a method of an object as an event handler

```
// Identify the node for which you want register event handler
var handlerNode = document.getElementById("handler");

// This connect() call ensures that when handlerNode.onclick()
//  is called, object.handler() will be called with the same arguments.
dojo.event.connect(handlerNode, "onclick", object, "handler");
```

Example #4: Registration of Multiple Handlers

```
var handlerNode = document.getElementById("handler");

// Connect also transparently handles multiple listeners.
// They are called in the order they are registered.
// This would kick off two separate actions from a single onclick event:
dojo.event.connect(handlerNode, "onclick", object, "handler");
dojo.event.connect(handlerNode, "onclick", object, "handler2");
```

Dojo Event System:
Chaining Function Calls

Attaching function of an object to another function

- Used when you have a need to invoke another function when a function is invoked
  - The source of the event is a function call not DOM event
- Use the same dojo.event.connect()
  - dojo.event.connect(srcObj, "srcFunc", targetObj, "targetFunc");

Example #5a: Attaching a function of an object to another function

```
// Define a simple object with a couple of methods
var exampleObj = {
    counter: 0,
    foo: function(){
        this.counter++;
        alert("foo: counter = " + this.counter);
    },
    bar: function(){
        this.counter++;
        alert("bar: counter = " + this.counter);
    }
};

// I want exampleObj.bar to get called whenever exampleObj.foo
// is called.  How can I do this?
```

Example #5b: Attaching a function of an object to another function

```
// We can set this up the same way that we do with DOM events.
// Now calling foo() will also call bar(), thereby incrementing the
// counter twice and alerting "foo" and then "bar".
dojo.event.connect(exampleObj, "foo", exampleObj, "bar");
```

Dojo Event System: Aspect-Oriented ("before advice" and "after advice")

Before Advice

- dojo.event.connect() can be used to call listeners (event handlers) before the source function is called
  - In Aspect Oriented Programming terminology, this is called "before advice"
- Example: "bar" gets called before "foo" when exampleObj.foo is called:
  - dojo.event.connect("before", exampleObj, "foo", exampleObj, "bar");
- “after advice” is a default

Example #6: Attaching a function of an object to another function - “before”

```
var exampleObj = {
    counter: 0,
    foo: function(){
        this.counter++;
        alert("foo: counter = " + this.counter);
    },
    bar: function(){
        this.counter++;
        alert("bar: counter = " + this.counter);
    }
};

// “bar” function will be called before “foo”
dojo.event.connect(“before”, exampleObj, "foo", exampleObj, "bar");
```

source: j2ee blueprints

Event Handler Wrapper

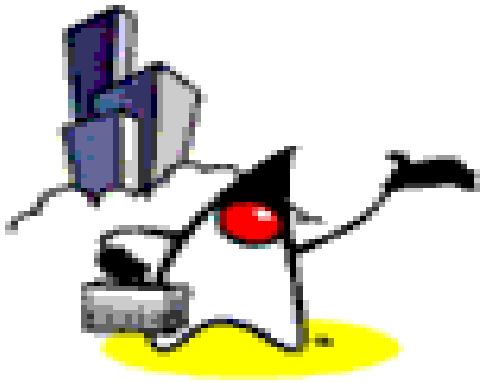
- In some case you may want to modify the behavior or arguments of an event handler without modifying the source code of the JavaScript component you are using
  - Adding "before" or "after" an event handler listeners may not be enough
- In Dojo, this is accomplished using dojo.io.connect with "around" as the first argument
  - dojo.io.connect(“around”, ...);
- It behaves like Servlet Filter scheme
  - Intercept requests and modify the behavior

Example #7: Event Handler Wrapper

```
 // custom event handler wrapper
function customLoadHandler(invocation) {
   alert("custom menu name =" + invocation.args[0].name);
   // update the name property of the argument
   invocation.args[0].name = "Custom Menu";
   //call the default event handler
   invocation.proceed();
}

function ImageScroller() {
    this.load = function (args) {
        alert("default menu name=" + args.name);
    }
}

var is = new ImageScroller();
dojo.event.connect("around", is, "load", this, "customLoadHandler");
is.load({name: "My Menu", items: ['File', 'Save']});
// alerts "custom menu name=My Menu"
// alerts "default menu name=Custom Menu"
```

Publish/Subscribe Event Handling

- Used when you have a need to communicate events anonymously between components
- Allows customizing the component to allow the topic name to be passed in as an initialization parameter to make the component more flexible



Demo: Dojo Event System

Dojo Event System: Disconnecting Event Handlers

Disconnecting Event Handler

- Use dojo.event.disconnect() for events
  - Must pass exactly the same arguments as were passed to dojo.event.connect()
- dojo.event.topic.unsubscribe() for topics
  - Must pass exactly the same arguments as were passed to dojo.event.topic.subscribe()

Dojo Event System:
Resources

Resources on Dojo Event System

- "Events for JavaScript Components" article written by Greg Murray
  - https://bpcatalog.dev.java.net/source/browse/*checkout*/bpcatalog/ee5/docs/ajax/handling-js-events.html
- "Dojo Event System" home page
  - http://dojotoolkit.org/docs/dojo_event_system.html
- "Dojo Event Examples" Wiki
  - http://dojo.jot.com/EventExamples

Dojo Widget

What is a Widget?

- Is a UI element such as a button, text box, scroll bar, calendar, tree etc
  - Widgets are a lot easier to deal with than DOM APIs
- Can be a composite element
- Can have custom style
- Event handlers can be registered
  - Event handlers might call backend service
- Can handle browser incompatibilities

Features of Dojo Widget System

- Ability to prototype, test, and tweak the component's UI and interactions
- Provides "templates" that you can use to rapidly prototype your UI in HTML and CSS and expose your widgets as markup on the pages you include them on
- Ensures that your work in prototyping isn't thrown away when it comes time for deployment

source: dojotoolkit.org

Mechanism

- Templates in Dojo place HTML and CSS fragments into files which are consulted when a widget is constructed
- The setup of event handlers and the creation of references to DOM nodes is handled through some extra attributes on your HTML markup

source: dojotoolkit.org

Dojo Widget:
Using Widgets

Three Ways of Adding Widgets to Your Page

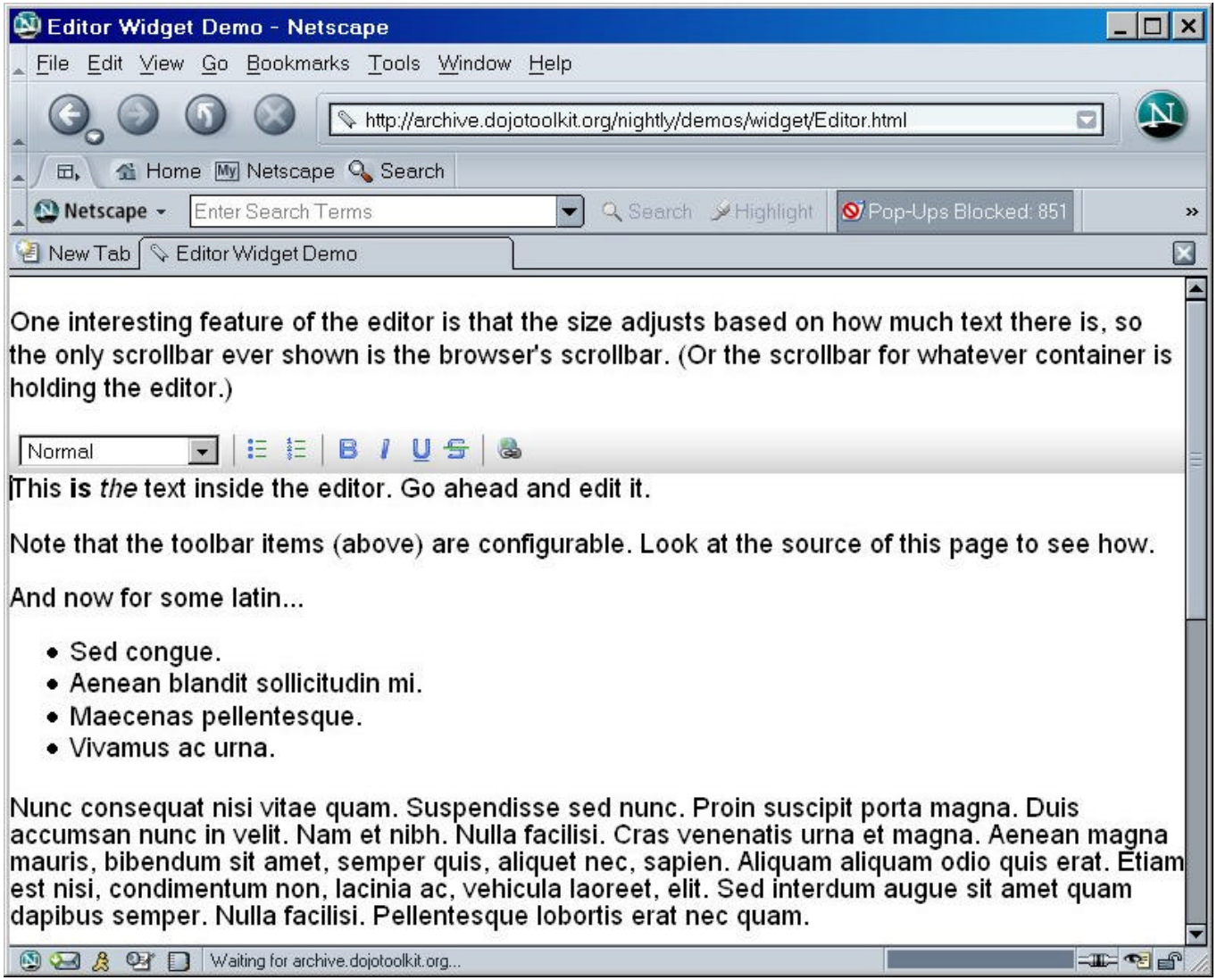
- You can use one of the following three methods
  - <dojo:NameofWidget ../>
  - <div dojoType="NameOfWidget" ..../>
  - <div class="dojo-NameOfWidget" .../>

Dojo Widget:
Built-in Widgets (from Dojo Toolkits v0.3 as of June, 2006)

Built-in Widgets from Dojo Toolkit (as of June 2006)

- You can use the built-in widgets just by adding them in your html or jsp pages
- List, Editor, Fisheye, Form, Mail, Tree, contentPane, datePicker, dialog, layoutContainer, rounded, spilitContainer, tabContainer, tooltip, etc.
- Many more are being created by community members

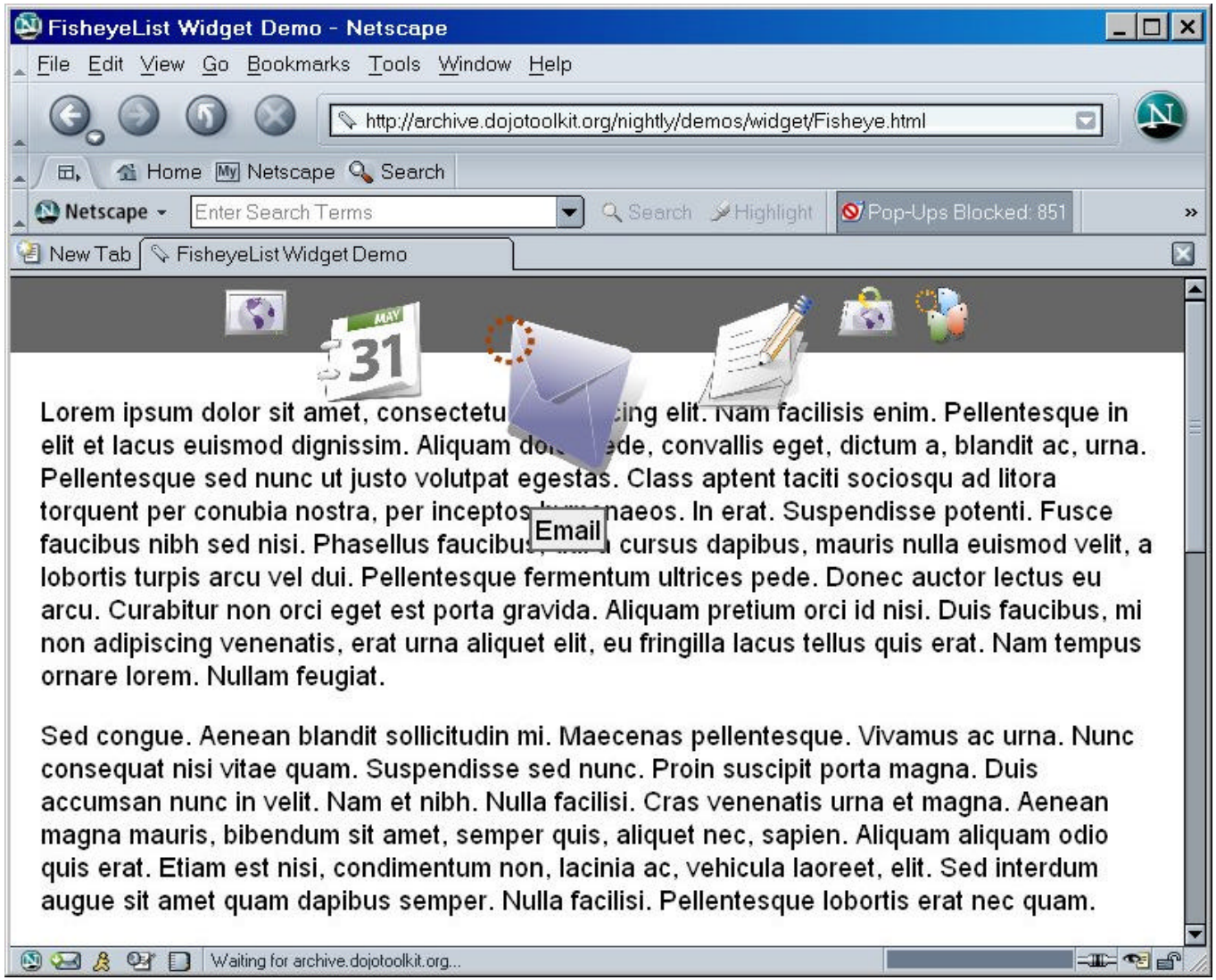
Editor Widget



How to use Editor Widget

```
<script>
    dojo.require("dojo.widget.Editor");
</script>

<body>
    <div dojoType="Editor" items="formatblock;|;
        insertunorderedlist;insertorderedlist;|
        ;bold;italic;underline;strikethrough;|;createLink;">
    ...
```

FisheyeList & FisheyeListItem Widgets



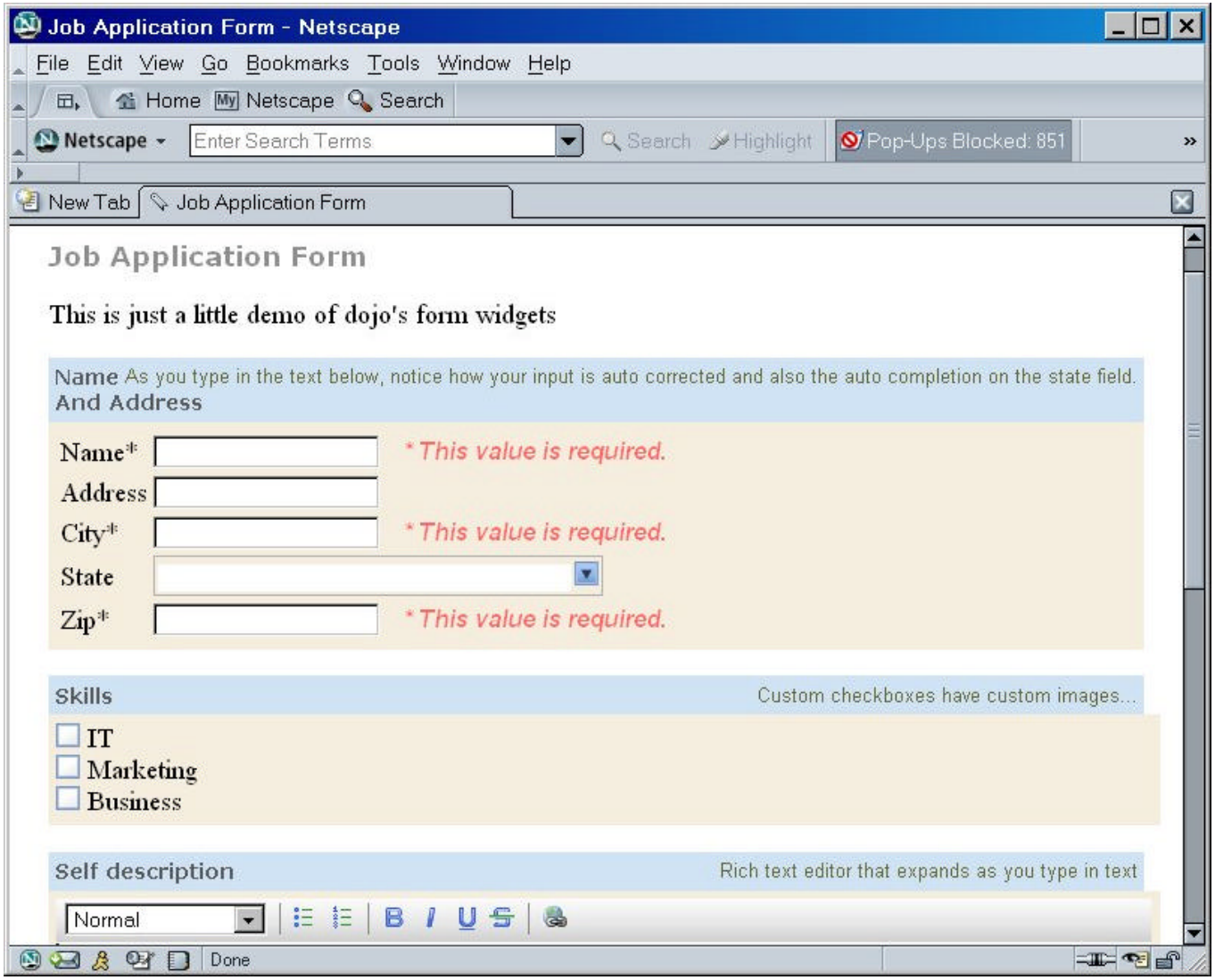
How to use FisheyeList & FisheyeListItem Widgets

```
<script>
    dojo.require("dojo.widget.FisheyeList");
    dojo.hostenv.writeIncludes();
</script>

<body>
    <div class="dojo-FisheyeList"
        dojo:itemWidth="50" dojo:itemHeight="50"
        dojo:itemMaxWidth="200" dojo:itemMaxHeight="200"
        dojo:orientation="horizontal"
        dojo:effectUnits="2"
        dojo:itemPadding="10"
        dojo:attachEdge="top"
        dojo:labelEdge="bottom"
        dojo:enableCrappySvgSupport="false"
        >
```

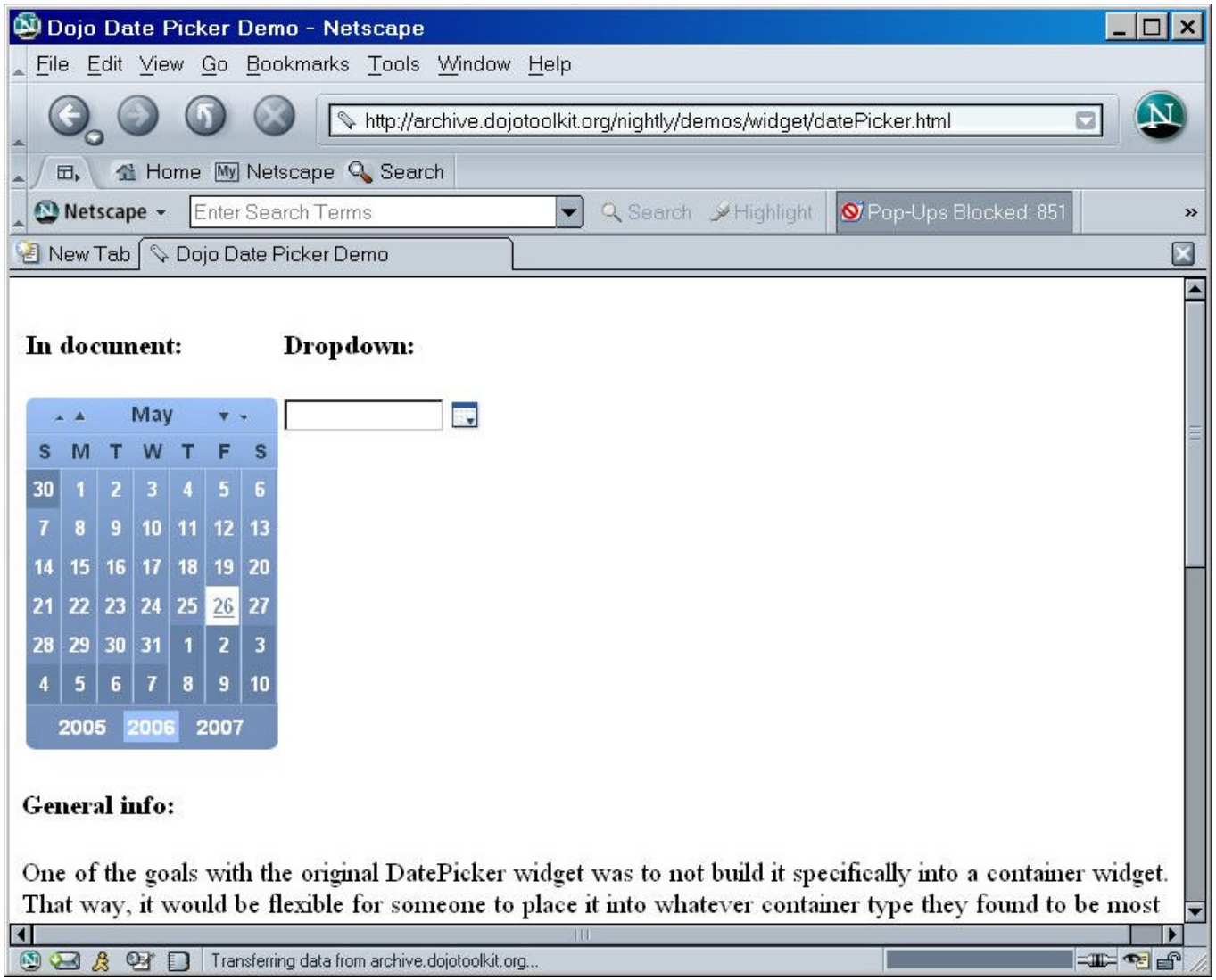
How to use FisheyeList & FisheyeListItem Widgets

```
<div class="dojo-FisheyeListItem" onClick="load_app(1);"
dojo:iconsrc="images/icon_browser.png" caption="Web Browser">
</div>

<div class="dojo-FisheyeListItem" onClick="load_app(2);"
dojo:iconsrc="images/icon_calendar.png" caption="Calendar">
</div>

<div class="dojo-FisheyeListItem" onClick="load_app(3);"
dojo:iconsrc="images/icon_email.png" caption="Email">
</div>

<div class="dojo-FisheyeListItem" onClick="load_app(4);"
dojo:iconsrc="images/icon_texteditor.png" caption="Text Editor">
</div>

...
```

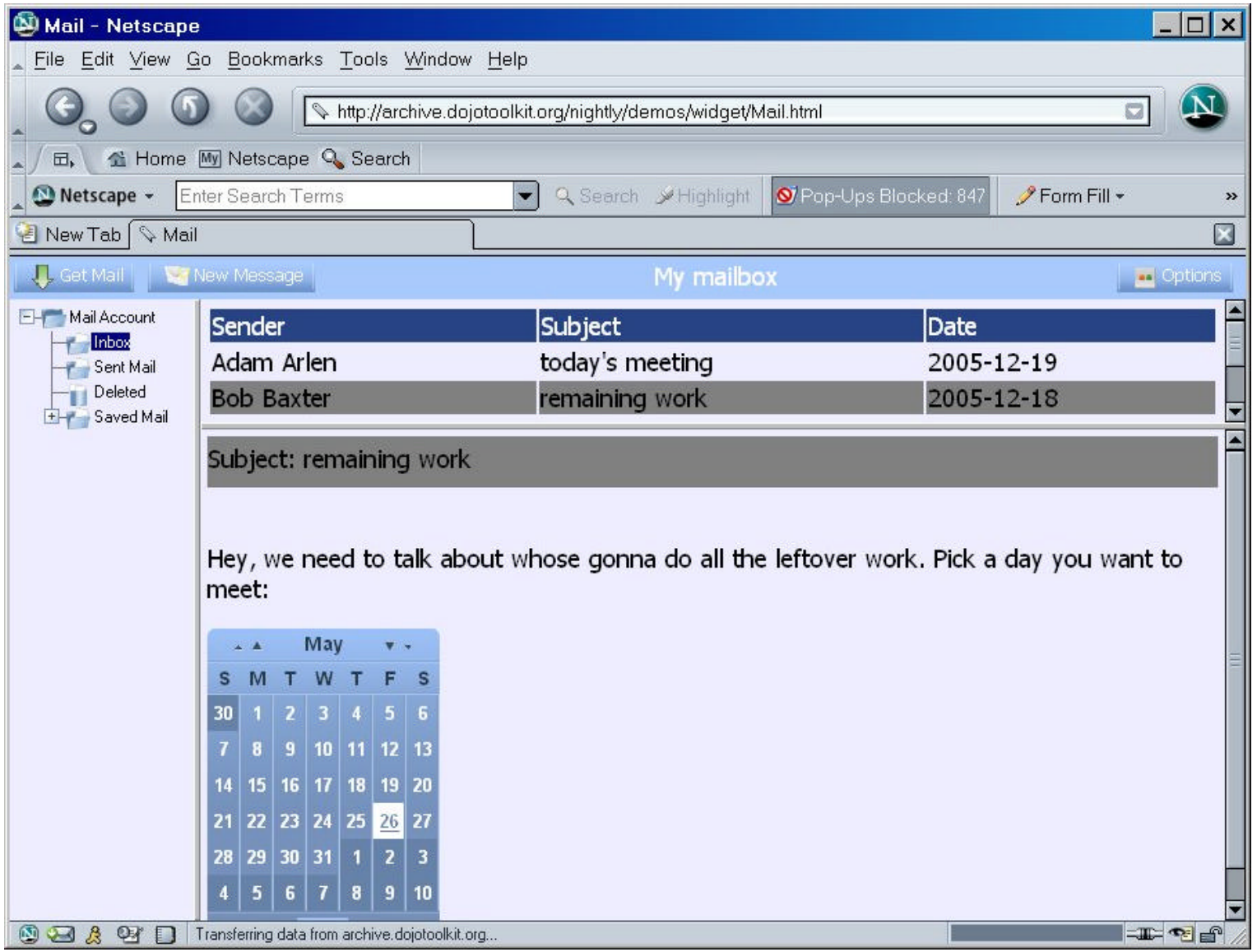
Form Widget


Job Application Form - Netscape
File Edit View Go Bookmarks Tools Window Help
Home Netscape Search
Netscape
Enter Search Terms
Search Highlight
Pop-Ups Blocked: 851
New Tab
Job Application Form
Job Application Form
This is just a little demo of dojo's form widgets
Name As you type in the text below, notice how your input is auto corrected and also the auto completion on the state field.
And Address
Name*
* This value is required.
Address
City*
* This value is required.
State
Zip*
* This value is required.
Skills
Custom checkboxes have custom images...
IT
Marketing
Business
Self description
Rich text editor that expands as you type in text
Normal
Done

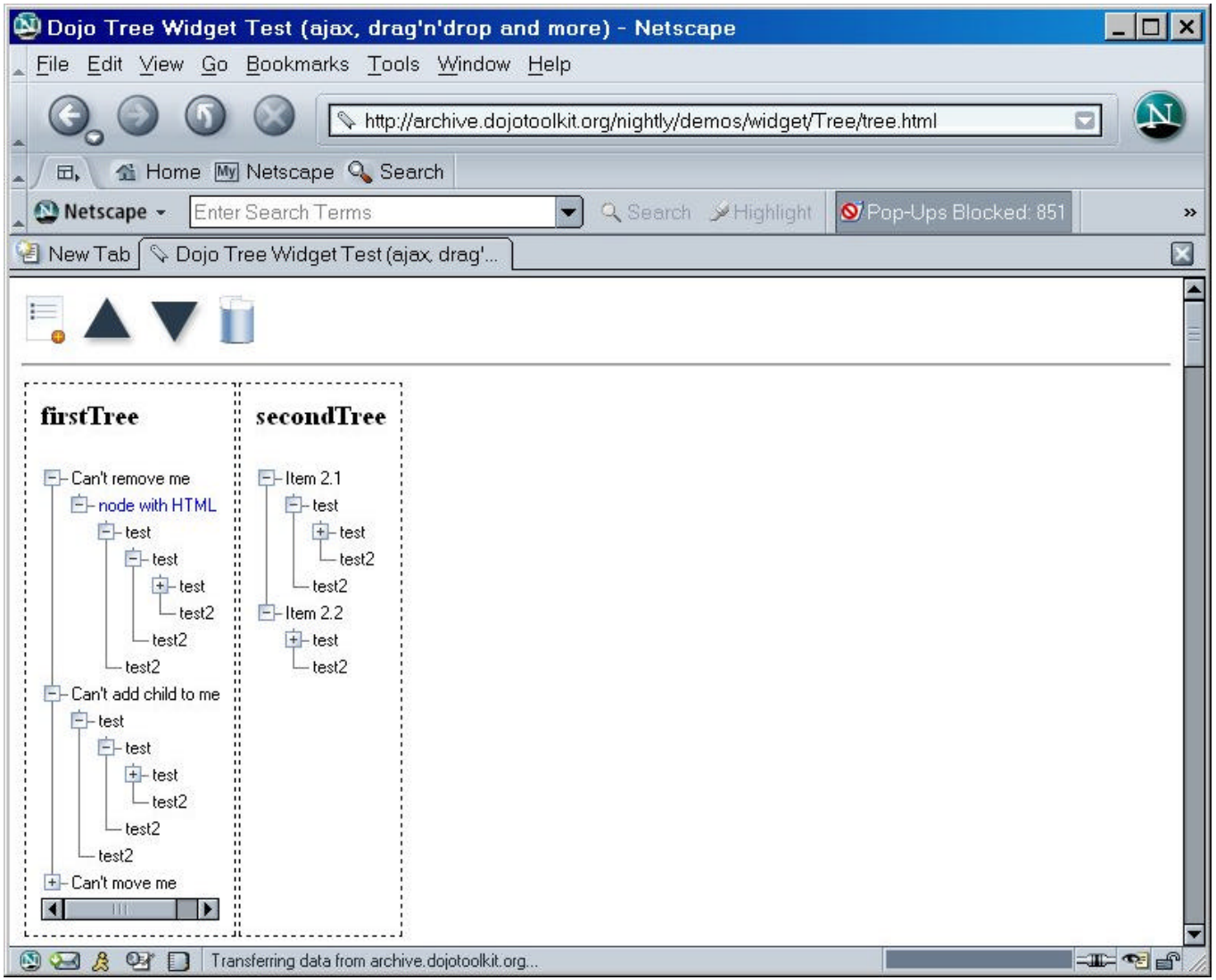
datePicker Widget

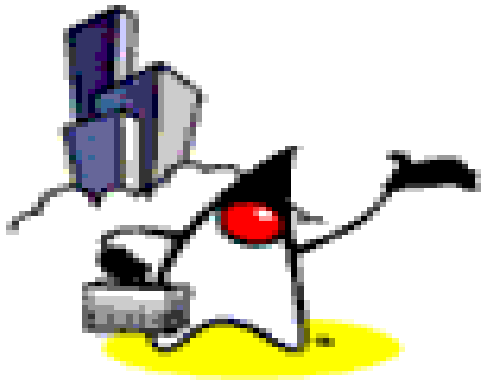


Mail Widget


Mail - Netscape
File Edit View Go Bookmarks Tools Window Help
http://archive.dojotoolkit.org/nightly/demos/widget/Mail.html
Home Netscape Search
Netscape Enter Search Terms Search Highlight Pop-Ups Blocked: 847 Form Fill
New Tab Mail
Get Mail New Message My mailbox Options
Mail Account
Inbox
Sent Mail
Deleted
Saved Mail
Sender Subject Date
Adam Arlen today's meeting 2005-12-19
Bob Baxter remaining work 2005-12-18
Subject: remaining work
Hey, we need to talk about whose gonna do all the leftover work. Pick a day you want to meet:
May
S M T W T F S
30 1 2 3 4 5 6
7 8 9 10 11 12 13
14 15 16 17 18 19 20
21 22 23 24 25 26 27
28 29 30 31 1 2 3
4 5 6 7 8 9 10
Transferring data from archive.dojotoolkit.org...

Tree Widget


Dojo Tree Widget Test (ajax, drag'n'drop and more) - Netscape
File Edit View Go Bookmarks Tools Window Help
http://archive.dojotoolkit.org/nightly/demos/widget/Tree/tree.html
Home Netscape Search
Netscape Enter Search Terms Search Highlight Pop-Ups Blocked: 851
New Tab Dojo Tree Widget Test (ajax, drag'...
firstTree
Can't remove me
node with HTML
test
test
test
test2
test2
test2
Can't add child to me
test
test
test
test2
test2
test2
Can't move me
secondTree
Item 2.1
test
test
test2
test2
Item 2.2
test
test2
Transferring data from archive.dojotoolkit.org...



Demo: Built-in Widgets from Dojo Toolkit

Demo Scenarios

- Accessing demo pages that contain various built-in Widgets from the web (or locally deployed demo pages if there is no internet)
  - http://archive.dojotoolkit.org/nightly/demos/widget
- See source (by clicking source button)

source: dojotoolkit.org

Introduction to Dojo Toolkit

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