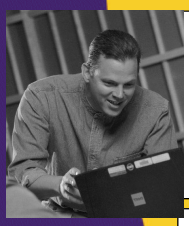
The Dojo JavaScript Toolkit
Part I: Ajax Support
(Dojo 1.3 Version)

Originals of Slides and Source Code for Examples:
http://courses.coreservlets.com/Course-Materials/ajax.html

© 2009 Marty Hall

For live Ajax & GWT training, see training courses at http://courses.coreservlets.com/.



Taught by the author of *Core Servlets and JSP*, *More Servlets and JSP*, and this tutorial. Available at public venues, or customized versions can be held on-site at your organization.

- Courses developed and taught by Marty Hall
  - Java 6, intermediate/beginning servlets/JSP, advanced servlets/JSP, Struts, JSF 1.*x* & 2.0, Ajax, GWT, custom mix of topics
  - Ajax courses can concentrate on 1 library (jQuery, Prototype/Scriptaculous, Ext-JS, Dojo, Google Closure) or survey several
- Courses developed and taught by coreservlets.com experts (edited by Marty)
  - Spring, Hibernate/JPA, EJB3, Ruby/Rails

Contact hall@coreservlets.com for details

Topics in This Section

- **Overview of Dojo**
- **Installation and documentation**
- **Quick summary of dojo.query and selectors**
- **The dojo.xhrGet function and variants**
  - Basics
  - Options
  - Sending data
  - Inserting HTML
  - Handling JSON data
- **Comparing Ajax support to other libraries**
  - Prototype, jQuery, Ext-JS

Introduction

Overview of Dojo

- **Core**
  - Utilities for Ajax, CSS querying, DOM manipulation, cross-browser event handling, and general JavaScript programming
  - *Very* similar to jQuery
- **Dijit**
  - Large set of rich GUI components
  - Much more extensive than jQuery UI
- **DojoX**
  - Charts, graphs, vector graphics, fancy widgets
  - Similar to the jQuery Plugins

Core Utilities

- **dojo.query("*css selector*")**
  - Returns a NodeList of matching DOM elements.
    - Almost identical to $("*css selector*") in jQuery.
- **dojo.xhrGet({*options*})**
  - Makes an Ajax GET request. Also dojo.xhrPost, etc.
    - Moderately similar to $.ajax({*options*}) in jQuery
  - Example
    - dojo.xhrGet({ url: "address", load: responseHandler });
- **dojo.addOnLoad(*function*)**
  - Calls function after DOM loaded.
    - Almost identical to $(*function*) in jQuery
- **dojo.require("dojo.*libraryName*")**
  - Loads Dojo libraries on the fly

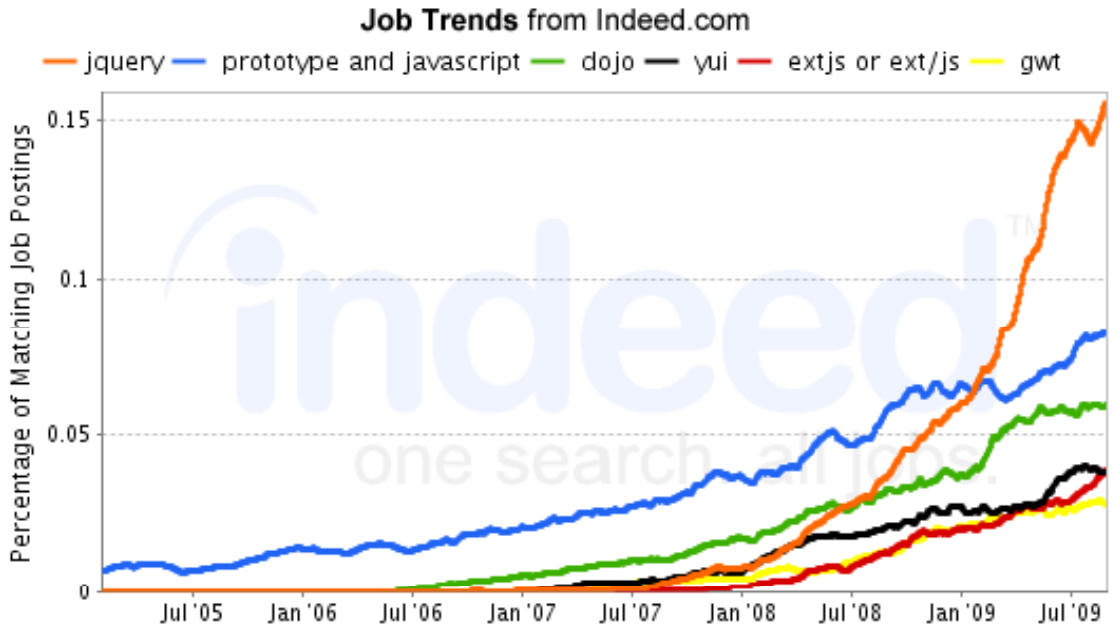
Downloading and Installation

- **Download**
  - http://download.dojotoolkit.org/
  - Choose "Download latest Stable Release", then dojo-release-x.y.z.zip or dojo-release-x.y.z.tar.gz
    - This tutorial uses release 1.3.1
- **Installation**
  - Unzip release file, creating 3 subfolders: dojo, dijit, dojox
    - Folders are quite large, but core file (dojo.js) is small.
    - Other files are loaded on-the-fly based on dojo.require statements.
  - Copy 3 folders to WebContent/scripts of Eclipse project
- **Online documentation**
  - http://dojotoolkit.org/docs

Browser Compatibility

- **Firefox**
  - Core: 1.5 or later (vs 2.0 for jQuery)
  - Dijit: 2.0 or later (same as jQuery)
- **Internet Explorer**
  - 6.0 or later (same as jQuery)
- **Safari**
  - 3.1 or later (vs. 3.0 for jQuery)
- **Opera**
  - Core: 9.6 or later (vs. 9.0 for jQuery)
  - Dijit: not supported!
- **Chrome**
  - 1.0 or later

Industry Usage



Approximately 40% of matches to "prototype and JavaScript" were false positives such as "build a prototype with JavaScript". So, discount the Prototype graph by about 40%.

dojo.query & Selectors: Basics

Note: brief intro only. More details in next tutorial section.

Selecting DOM Elements

- **Idea**
  - Use dojo.query("css selector") to get a set of DOM elements
    - Then, perform operations on each (see next page)
- **Examples**
  - dojo.query("#some-id")
    - Return 1-element set (or empty set) of element with id
    - Simplest use, and most common for Ajax (note the "#"!)
    - Can also use dojo.byId("some-id") to get single element
  - dojo.query("p")
    - Return all p elements
  - dojo.query(".blah")
    - Return all elements that have class="blah"
  - dojo.query("li b span.blah")
    - Return all elements that are inside b elements, that in turn are inside li elements

Manipulating DOM Elements

- **Common functions on matched elements**
  - dojo.query("selector").forEach(function)
    - Calls function on each element. "this" set to element.
  - dojo.query("selector").addClass("name")
    - Adds CSS class name to each. Also removeClass, toggleClass
  - dojo.query("selector").wipeOut().play()
    - Makes invisible. Also wipeIn, fadeOut, fadeIn, etc.
  - dojo.query("selector").onclick(function)
    - Adds onclick handler. Also onchange, onmouseover, etc.
  - dojo.query("selector").html("<tag>some html</tag>")
    - Sets the innerHTML of each element.
      - Must use dojo.require("dojo.NodeList-html");
  - dojo.byId("some-id").value
    - Returns value of input element. Notice no "#".
- **Chaining**
  - dojo.query("a").onclick(f1).addClass("name").forEach(f2)

Example: Randomizing Background Colors (JavaScript)

```
dojo.require("dojo.NodeList-fx");

dojo.addOnLoad(function() {
    dojo.query("#button1").onclick(randomizeHeadings);
    dojo.query("#button2").onclick(revertHeadings);
});
```

Loads extra JavaScript on the fly.

The main two needed for basics apps are dojo.NodeList-fx and dojo.NodeList-html. The online docs tell you if a certain function necessitates a "dojo.require" statement.

Like smart window.onload. Explained in next section.

Sets onclick handlers

Example: Randomizing Colors (JavaScript Continued)

```
function randomizeHeadings() {
  dojo.query("h3").forEach(setRandomStyle);
  dojo.query("h3.green").wipeOut().play();
}

function setRandomStyle(heading) {
  dojo.query(heading).addClass(randomStyle());
}
```

Call setRandomStyle function on each h3 element

Slowly hide every h3 that has CSS style "green". Note difference from jQuery approach: wipeOut returns an Animation, which you then call play on.

Add "red", "yellow" or "green" CSS names to each

Example: Randomizing Colors (JavaScript Continued)

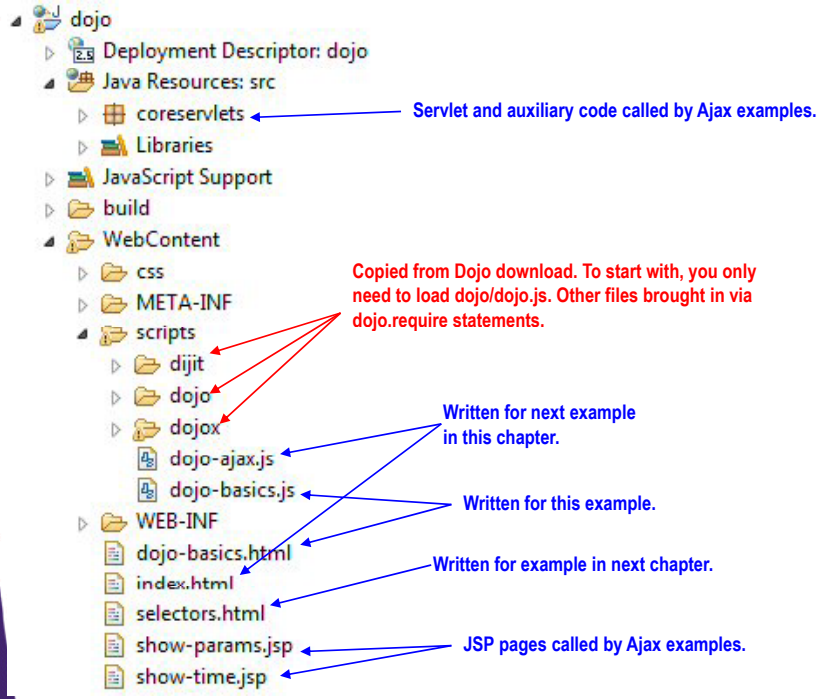
```
function randomStyle() {
  var styles = ["red", "yellow", "green"];
  return(randomElement(styles));
}

function randomElement(array) {
  var index = Math.floor(Math.random()*array.length);
  return(array[index]);
}

function revertHeadings() {
  dojo.query("h3.green").wipeIn().play();
  dojo.query("h3").removeClass("red")
                  .removeClass("yellow")
                  .removeClass("green");
}
```


Example: Randomizing Colors (Style Sheet)

```
.red { background-color: red }
.yellow { background-color: yellow }
.green { background-color: green }

...
```

Names set by setRandomStyles function

Example: Eclipse Setup



Example: Randomizing Colors (HTML)

```
...
<head><title>Dojo Basics</title>
<link rel="stylesheet"
      href="./css/styles.css"
      type="text/css"/>
<script src="./scripts/dojo/dojo.js"
        type="text/javascript"></script>
<script src="./scripts/dojo-basics.js"
        type="text/javascript"></script>
</head>
```

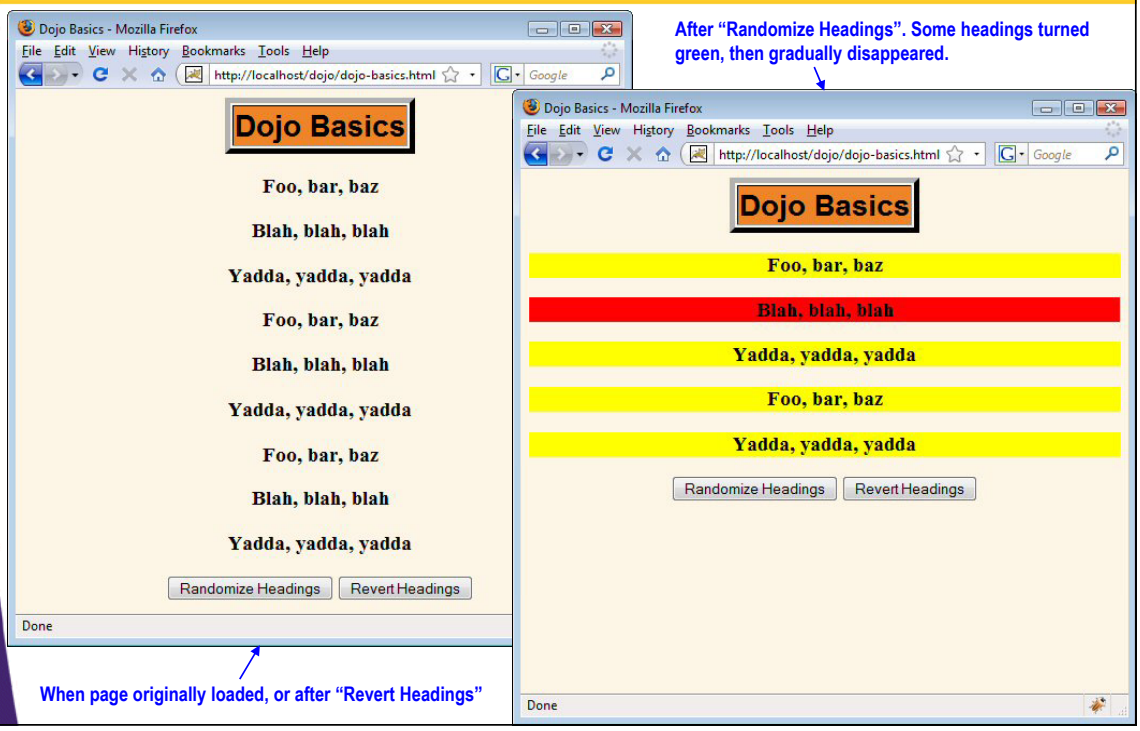
Core Dojo. The dojo.require statements cause on-the-fly loading of additional files.

Example: Randomizing Colors (HTML Continued)

```
...
<h3>Foo, bar, baz</h3>
<h3>Blah, blah, blah</h3>
<h3>Yadda, yadda, yadda</h3>
<h3>Foo, bar, baz</h3>
<h3>Blah, blah, blah</h3>
<h3>Yadda, yadda, yadda</h3>
<h3>Foo, bar, baz</h3>
<h3>Blah, blah, blah</h3>
<h3>Yadda, yadda, yadda</h3>
<form action="#">
  <input type="button" id="button1"
         value="Randomize Headings"/>
  <input type="button" id="button2"
         value="Revert Headings"/>
</form> ...
```

The ids to which onclick handlers were attached.

Example: Randomizing Colors (Results)



Understanding Operations on Sets of Elements

- **Instead of this**

```
function randomizeHeadings() {
   dojo.query("h3").forEach(setRandomStyle);
   dojo.query("h3.green").wipeOut().play();
}
function setRandomStyle(heading) {
   dojo.query(heading).addClass(randomStyle());
}
```

- **Why can't I simply do this?**

```
function randomizeHeadings() {
   dojo.query("h3").addClass(randomStyle());
   dojo.query("h3.green").wipeOut().play();
}
```


dojo.xhrGet: Basics

dojo.xhrGet: Basic Syntax

- **dojo.xhrGet(optionsObject)**
  - Minimal form: dojo.xhrGet({url: "address", load: funct});
    - Also dojo.xhrPost, dojo.xhrPut, dojo.xhrDelete, etc.
  - Handler function gets response text. Response text is considered a string unless you use handleAs option.
- **Options for dojo.xhrGet({…})**
  - Almost-always used
    - url, load
  - Other common options
    - content, error, preventCache, handleAs

Data-Centric Ajax with and without Toolkits

- **With basic JavaScript**

```
function getRequestObject() {
  if (window.XMLHttpRequest) {
    return(new XMLHttpRequest());
  } else if (window.ActiveXObject) {
    return(new ActiveXObject("Microsoft.XMLHTTP"));
  } else { return(null); }
}
function sendRequest() {
  var request = getRequestObject();
  request.onreadystatechange =
    function() { someFunct(request); };
  request.open("GET", "some-url", true);
  request.send(null);
}
```

Data-Centric Ajax with and without Toolkits

- jQuery (handler passed response text)

```
$.ajax({url: "address",
        success: handlerFunct});
```

- Prototype (handler passed response object)

```
new Ajax.Request("address",
                 {onSuccess: handlerFunct});
```

- Ext (handler passed response object)

```
Ext.Ajax.request({url: "address",
                  success: handlerFunct});
```

- Dojo (handler passed response text)

```
dojo.xhrGet({url: "address",
             load: handlerFunct});
```


dojo.xhrGet Example Code: JavaScript

```
function showTime1() {
  dojo.xhrGet({
    url: "show-time.jsp",
    load: showAlert,
    preventCache: true
  });
}

function showAlert(text) {
  alert(text);
}
```

The preventCache option is not required, but is a convenient option when the same URL (including query data) yields different responses. This way, you don't have to send Cache-Control and Pragma headers from server.

This is the response text, but you can declare a second argument (ioArgs) if you want the response object (XmlHttpRequest) and other information. Also note that the latest Firefox does not let you pass native functions here, so you cannot use alert instead of showAlert for the success parameter.

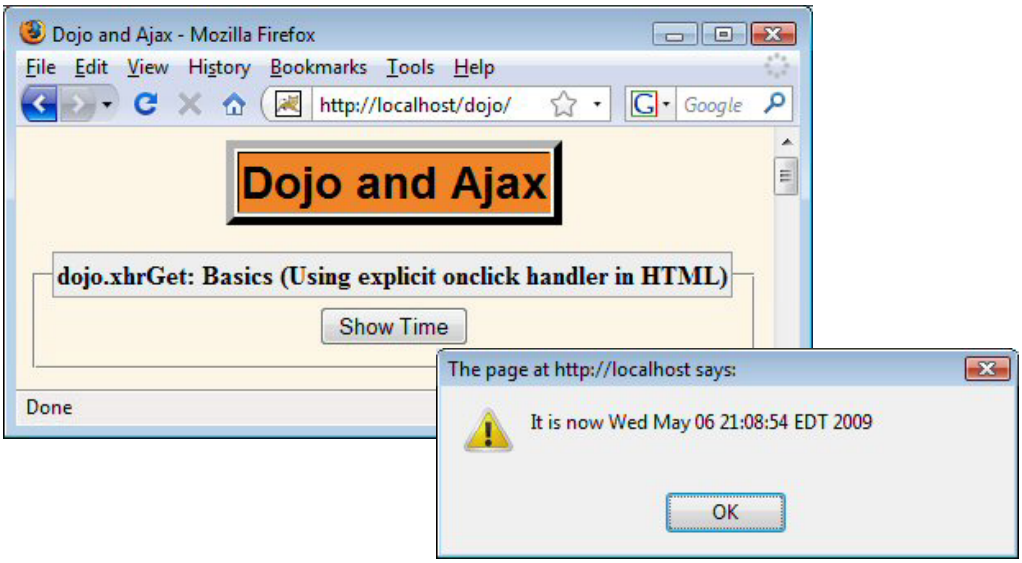
dojo.xhrGet Example Code: HTML

```
…
<head><title>Dojo and Ajax</title>...
<script src="./scripts/dojo/dojo.js"
        type="text/javascript"></script>
<script src="./scripts/dojo-ajax.js"
        type="text/javascript"></script>
</head>
<body>...
<fieldset>
  <legend>dojo.xhrGet: Basics (Using explicit
          onclick handler in HTML)</legend>
  <form action="#">
    <input type="button" value="Show Time"
           onclick="showTime1()"/>
  </form>
</fieldset>…
```


dojo.xhrGet Example Code: JSP

```
It is now <%= new java.util.Date() %>
```

dojo.xhrGet : Results

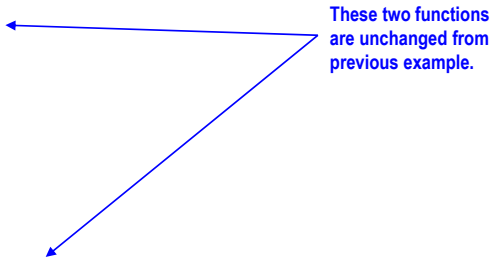


Registering Event Handlers in JavaScript

- **Basic approach**
  - Previous example set the onclick handler in the HTML. Although this is common with other Ajax libraries, Dojo (and jQuery) advocate setting it in the JavaScript instead
    - Often referred to as "unobtrusive JavaScript": no explicit JavaScript anywhere in the HTML page
- **Dojo support**
  - dojo.addOnLoad(function() {…});
    - Function runs after the DOM is loaded, but does not wait for images, as with window.onload
    - Use this approach to set up *all* event handlers
  - dojo.query("#some-id").onclick(someHandler);
    - Assigns onclick handler. Handler is passed a DOMEvent with characteristics that are unified across browsers

Redoing Time Alert: JavaScript

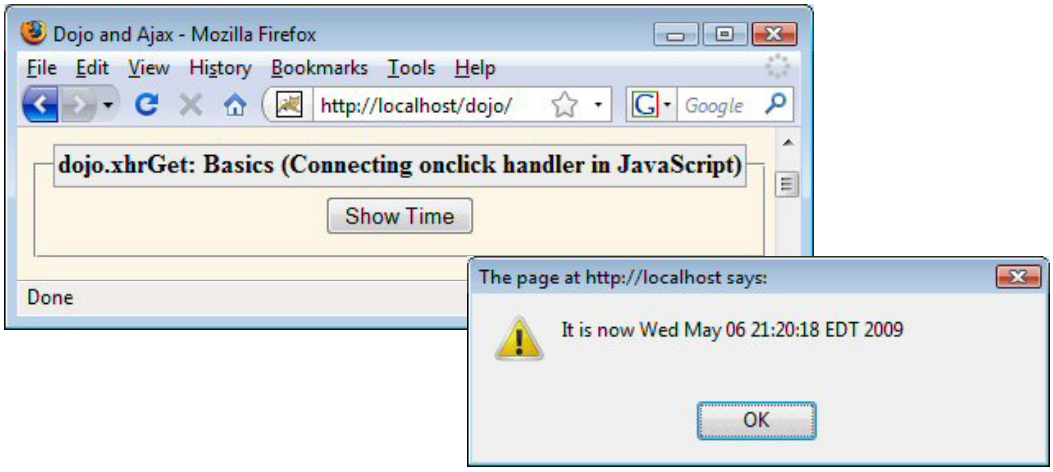
```
dojo.addOnLoad(function() {
  dojo.query("#time-button-1").onclick(showTime1);
});

function showTime1() {
  dojo.xhrGet({
    url: "show-time.jsp",
    load: showAlert,
    preventCache: true
  });
}

function showAlert(text) {
  alert(text);
}
```



Redoing Time Alert: HTML

```
<fieldset>
  <legend>dojo.xhrGet: Basics (Registering
          onclick handler in JavaScript)</legend>
  <form action="#">
    <input type="button" value="Show Time"
           id="time-button-1"/>
  </form>
</fieldset>
```

Redoing Time Alert: Results



Works exactly the same as previous example.

dojo.xhrGet: Sending Data

Ajax Functions

- **dojo.xhrGet({options})**
  - Make a GET request
- **dojo.xhrPost({options})**
  - Make a POST request
- **dojo.xhr("method", {options}, hasBody)**
  - Lets you send arbitrary HTTP commands
- **dojo.xhrPut, dojo.xhrDelete**
  - Make a PUT or DELETE request
  - Unsupported on many servers
- **dojo.rawXhrPost, dojo.rawXhrPut**
  - Sends a POST or PUT request, but lets you provide the raw data for the body of the request

Overview

- **dojo.xhrGet({ url: …, load: …, content: …});**
  - Content value is an object: query string gets built out of property names and URL-encoded property values
    - On end of URL for ajax.xhrGet; in POST body for dojo.xhrPost
    - See later example for building the string automatically using the "form" option
    - Works identically to "parameters" option in Prototype or "data" option in jQuery
- **Examples**
  - dojo.xhrGet({… content: { param1: "foo bar!", param2: "baz"}});

Content Example: JavaScript

```
dojo.addOnLoad(function() {
  dojo.query("#params-button-1").onclick(showParams1);
  …
});

function showAlert(text) {
  alert(text);
}

function showParams1() {
  dojo.xhrGet({
    url: "show-params.jsp",
    content: { param1: "foo",
               param2: "bar" },
    load: showAlert
  });
}
```

Same function used in earlier examples.

The preventCache option is not used since the same data always results in the same response.

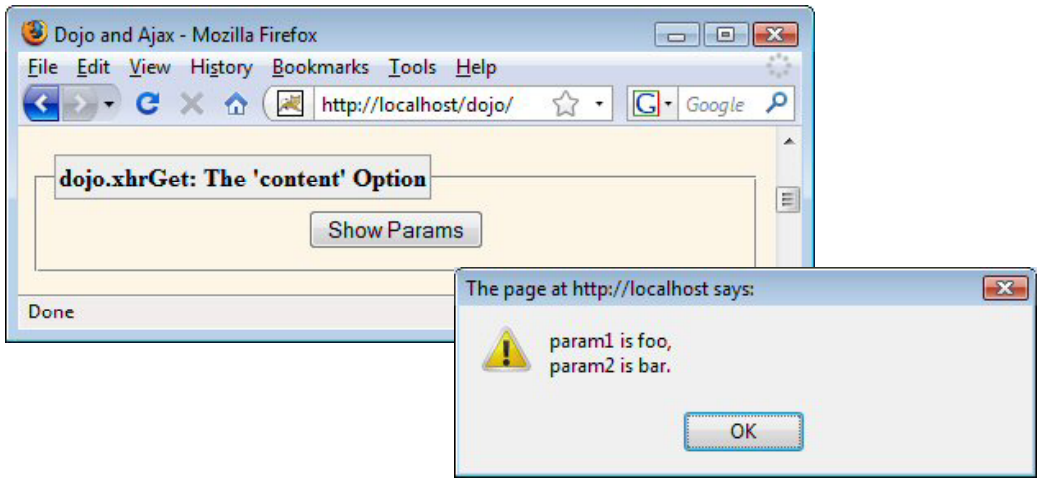
Content Example: HTML

```
…
<fieldset>
  <legend>dojo.xhrGet: The 'content' Option</legend>
  <form action="#">
    <input type="button" value="Show Params"
           id="params-button-1"/>
  </form>
</fieldset>
…
```

Content Example: JSP

```
param1 is ${param.param1},
param2 is ${param.param2}.
```


Content Example: Results



dojo.xhrGet: Options

Overview

- **Options (almost) always used: url, load**
  - dojo.xhrGet({url: "some-address", load: someFunc});
    - load is not strictly required; you might want to just fire off some data to the server and not display anything
    - Options apply equally to dojo.xhrPost and others
- **Common options: example**

```
dojo.xhrGet({
  url: "address",
  load: successHandlerFunction,
  content: { param1: "foo bar", param2: "baz"},
  error: errorHandlerFunction,
  preventCache: true,
  handleAs: "json" });
```

Options

Name	Description	Default
content	Data to send to server, in the form on an object with param names and raw (non-escaped) param values. The object property names become request param names and property values get URL-encoded and become request param values. & and = inserted automatically. Sent in the appropriate place depending on whether it is GET or POST.	*Empty*
error	Function to be called if request fails due to server error, expired timeout, or exception thrown from main response handler. Function gets passed 2 args: the data returned from the server (formatted according to the handleAs property), and an ioArgs object. The ioArgs object has many properties, but most useful are args, query, url, and xhr.	*None*
form	Id of a form whose fields will be serialized to form the query data.	*None*
handle	Function to be called whether or not request was successful. More common to use separate load and error properties.	*None*
handleAs	The format in which to pass the response to the handler function. Legal values are text, json, xml, json-comment-optional, json-comment-filtered, and javascript.	"text"

Options (Continued)

Name	Description	Default
headers	Extra HTTP request headers to be sent to the server.	*None*
load	Function to be called if request succeeds. Function gets passed 2 args: the data returned from the server (formatted according to the handleAs property), and an ioArgs object. The ioArgs object has many properties, but most useful are args, query, url, and xhr.	*None*
preventCache	Is browser prohibited from caching the page? Set to true if you use GET and you could get different responses back from the same data. If true, then the URL is appended with a dojo.preventCache parameter with a timestamp that changes with each request.	false
sync	Should the request be synchronous? Use synchronous requests with caution since they lock up the browser.	false
timeout	Timeout in milliseconds. If request takes longer, the error handler will be called instead of the load handler.	Infinity
url	The address to request. Should be a relative URL.	*None*

Inserting Results into HTML: The "html" Function

Content-Centric Ajax with and without Toolkits

- **jQuery**

```
function ajaxResult(address, resultRegion) {
  $(resultRegion).load(address);
}
```

- **Prototype**

```
function ajaxResult(address, resultRegion) {
  new Ajax.Updater(resultRegion, address);
}
```

- **Dojo**
  - *No explicit support for content-centric Ajax. Response handler uses "html" to do the insert.*
- **Ext-JS**

```
function ajaxResult(address, resultRegion) {
  Ext.get(resultRegion).load({ url: address});
}
```

Reading Textfield Values

1. **Use dojo.byId("some-id").value**
   - Note that there is no # before the ID, since you are not doing a CSS match, but rather using a shortcut for document.getElementById
   - Also note that Dojo has no way of looking up values directly from a NodeList resulting from a Dojo query. I.e., nothing equivalent to jQuery's $("#some-id").val().
2. **Supply values to "content" property**
   - No need to escape values first
     - dojo.xhrGet({… content: { param1: dojo.byId("id1").value });
- **Shortcut**
  - See upcoming section on "form" property

Ajax with HTML Insertion: JavaScript (Setup)

```
dojo.require("dojo.NodeList-html");

dojo.addOnLoad(function() {
  dojo.query("#params-button-2")
      .onclick(showParams2);
  …
});
```

Ajax with HTML Insertion: JavaScript (Main Code)

```
function showParams2() {
  dojo.xhrGet({
    url: "show-params.jsp",
    content: { param1: dojo.byId("field1").value,
               param2: dojo.byId("field2").value },
    load: function(text) {
            insertText(text, "#result1");
          }
  });
}

function insertText(text, selector) {
  dojo.query(selector).html(text);
}
```

field1 and field2 are textfield ids (not names)

id of div whose innerHTML will become the result text.

HTML Insertion: HTML Page

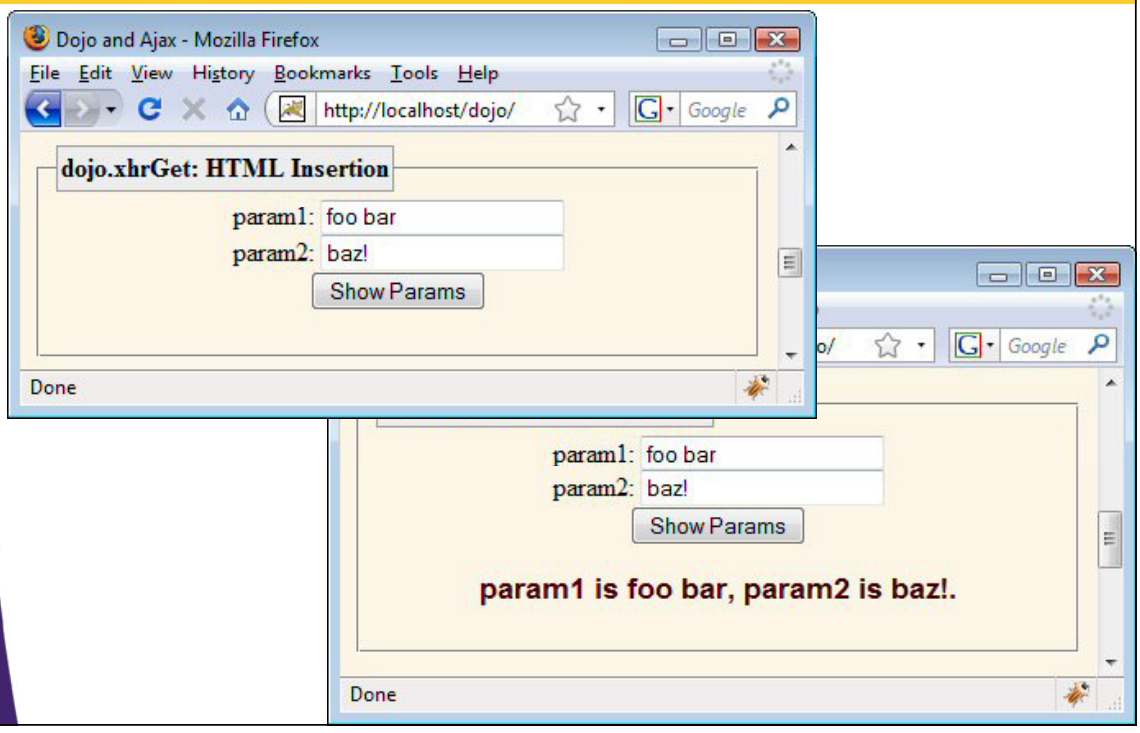
```
…
<fieldset>
  <legend>dojo.xhrGet: HTML Insertion</legend>
  <form action="#">
    param1:
    <input type="text" id="field1"/>
    <br/>
    param2:
    <input type="text" id="field2"/>
    <br/>
    <input type="button" value="Show Params"
           id="params-button-2"/>
    <h2 id="result1"></h2>
  </form>
</fieldset>
…
```

HTML Insertion: JSP

```
param1 is ${param.param1},
param2 is ${param.param2}.
```

Unchanged from previous example

HTML Insertion: Results



HTML Insertion: Comparing Prototype, jQuery, and Dojo

- **Prototype**

```
function ajaxResult(address, resultRegion) {
  new Ajax.Updater(resultRegion, address); }
```

- **jQuery**

```
function ajaxResult(address, resultRegion) {
  $(resultRegion).load(address); }
```

- **Dojo (also need dojo.require)**

```
function ajaxResult(address, resultRegion)
  dojo.xhrGet({ url: address,
                load: function(text) {
                  insertText(text, resultRegion);
                } }); }
function insertText(text, selector) {
  dojo.query(selector).html(text); }
```


Building Parameter Strings Automatically with "form"

Using "form" Option

- **Idea**
  - You specify a form, and Dojo automatically builds query string from all appropriate input elements.
  - The element names (not ids) become the param names
- **Syntax**
  - dojo.xhrGet({ url: …, load: …, form: "some-id" });
    - Note: no # before the id
- **Advantages**
  - One simple option, no matter how many input elements
  - Only takes values of active elements (e.g., unchecked radio buttons or checkboxes are ignored)
  - Giving names to input elements is familiar to HTML developers

"form" Example: JavaScript

```
dojo.require("dojo.NodeList-html");

dojo.addOnLoad(function() {
  dojo.query("#params-button-3")
      .onclick(showParams3);
  …
});

function showParams3() {
  dojo.xhrGet({
    url: "show-params.jsp",
    form: "form1",
    load: function(text) {
            insertText(text, "#result2");
          }
  });
} …
```

"form" Example: HTML

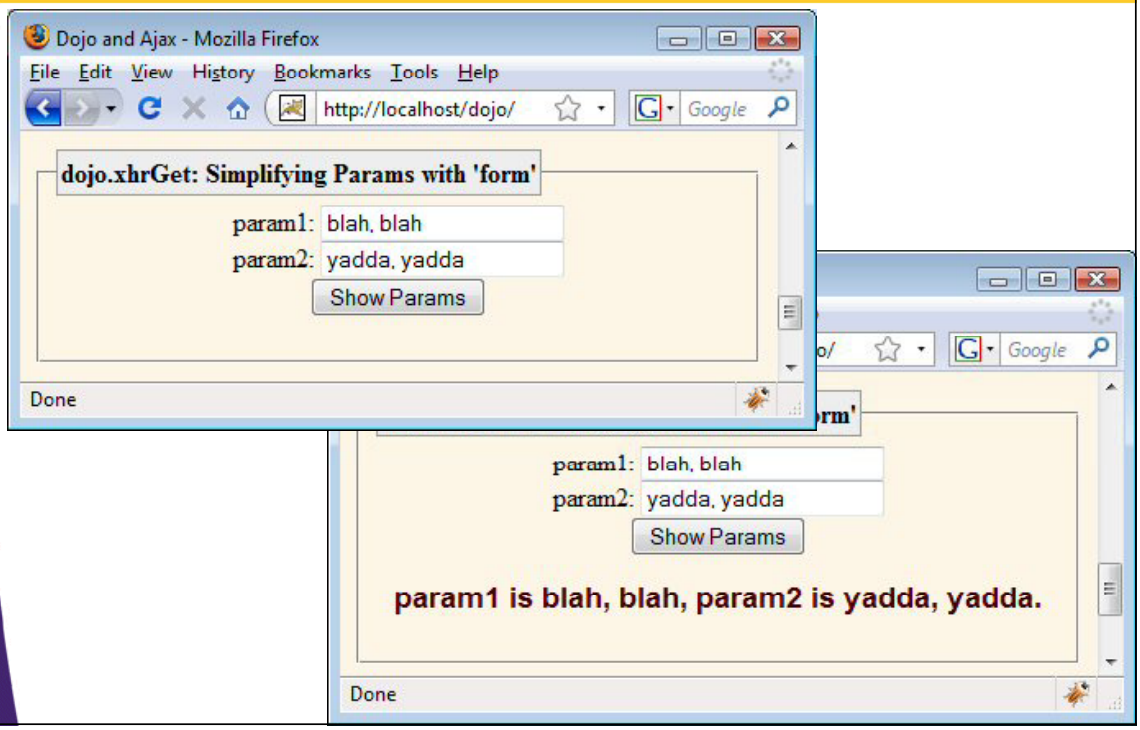
```
…
<fieldset>
  <legend>dojo.xhrGet: Simplifying Params
          with 'form'</legend>
  <form action="#" id="form1">
    param1:
    <input type="text" name="param1"/>
    <br/>
    param2:
    <input type="text" name="param2"/>
    <br/>
    <input type="button" value="Show Params"
           id="params-button-3"/>
    <h2 id="result2"></h2>
  </form>
</fieldset> …
```


"form" Example: JSP

```
param1 is ${param.param1},
param2 is ${param.param2}.
```

Unchanged from previous examples

"form" Example: Results



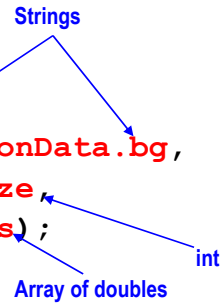
Handling JSON Data

Approach

- **Server**
  - Returns JSON object with no extra parens. E.g.:
    - { cto: "Resig ", ceo: "Gates ", coo: "Ellison" }
- **Code that calls dojo.xhrGet**
  - Specifies "json" for handleAs. E.g.:
    - dojo.xhrGet({url: address, load: handler, handleAs: "json"});
- **Response handler**
  - Receives JavaScript data as first argument. No need for parsing or "eval". Must build HTML from result. E.g.:
    - function handler(companyExecutives) {
      dojo.query("#some-id")
      .html("Chief Technology Officer is " +
      companyExecutives.cto + "");
      }

JSON Example Code: Core JavaScript

```
dojo.require("dojo.NodeList-html");
dojo.addOnLoad(…);

function showNums() {
  dojo.xhrGet({
    url: "show-nums",
    handleAs: "json",
    load: showNumberList,
    preventCache: true
  });
}

function showNumberList(jsonData) {
  var list = makeList(jsonData.fg, jsonData.bg,
                      jsonData.fontSize,
                      jsonData.numbers);
  dojo.query("#result3").html(list);
}
```



JSON Example Code: Auxiliary JavaScript

```
function makeList(fg, bg, fontSize, nums) {
  return(
    listStartTags(fg, bg, fontSize) +
    listItems(nums) +
    listEndTags());
}

function listStartTags(fg, bg, fontSize) {
  return(
    "<div style='color:" + fg + "; " +
                "background-color:" + bg + "; " +
                "font-size:" + fontSize + "px'>\n" +
    "<ul>\n");
}
```


JSON Example Code: Auxiliary JavaScript (Continued)

```
function listItems(items) {
  var result = "";
  for(var i=0; i<items.length; i++) {
    result = result + "<li>" + items[i] + "</li>\n";
  }
  return(result);
}

function listEndTags() {
  return("</ul></div>");
}
```

JSON Example Code: HTML

```
…
<fieldset>
  <legend>dojo.xhrGet: Treating Response
          as JSON</legend>
  <form action="#">
    <input type="button" value="Show Nums"
           id="nums-button"/>
    <div id="result3"></div>
  </form>
</fieldset>
…
```


JSON Example Code: Servlet

```
public class ShowNumbers extends HttpServlet {
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
      throws ServletException, IOException {
    response.setHeader("Cache-Control", "no-cache");
    response.setHeader("Pragma", "no-cache");
    String fg = ColorUtils.randomColor();
    request.setAttribute("fg", fg);
    String bg = ColorUtils.randomColor();
    request.setAttribute("bg", bg);
    String fontSize = "" + (10 + ColorUtils.randomInt(30));
    request.setAttribute("fontSize", fontSize);
    double[] nums =
      { Math.random(), Math.random(), Math.random() };
    request.setAttribute("nums", nums);
    response.setContentType("application/json");
    String outputPage = "/WEB-INF/results/show-nums.jsp";
    RequestDispatcher dispatcher =
      request.getRequestDispatcher(outputPage);
    dispatcher.include(request, response);
}}
```

JSON Example Code: JSP

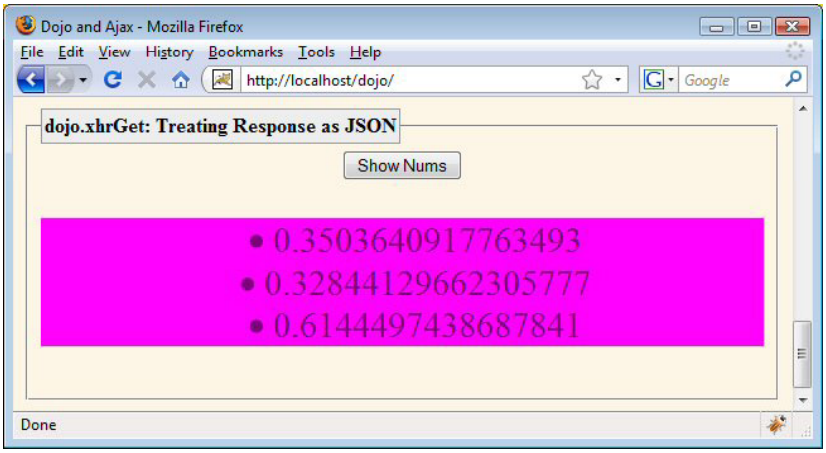
```
{ fg: "${fg}",
  bg: "${bg}",
  fontSize: ${fontSize},
  numbers: [ ${nums[0]}, ${nums[1]}, ${nums[2]}]
}
```

- **Notes**
  - No enclosing parens. Dojo will wrap in parens and then pass to "eval" (or equivalent functionality)
  - Types
    - fg and bg: Strings
    - fontSize: int
    - numbers: Array of doubles

JSON Example Code: Auxiliary Java Code

```
public class ColorUtils {
  private static String[] colors = {
    "aqua", "black", "blue", "fuchsia", "gray",
    "green", "lime", "maroon", "navy", "olive",
    "purple", "red", "silver", "teal", "white", "yellow"
  };

  /** A random number between 0 and range-1, inclusive. */

  public static int randomInt(int range) {
    return(new Random().nextInt(range));
  }

  /** One of the official HTML color names, at random. */

  public static String randomColor() {
    return(colors[randomInt(colors.length)]);
  }
}
```

JSON Example: Results



Wrap-up

"Best" JavaScript Libraries

- **General JS programming**
  - Leader: Prototype
- **Other programming (Java)**
  - Leader (only): GWT
- **DOM manipulation**
  - Leader: jQuery
    - Others copying jQuery approach and closing gap. jQuery released CSS matching library separately (http://sizzlejs.com)
- **Rich GUIs**
  - Leaders: Ext-JS, YUI, Dojo
  - 2nd tier: jQuery UI, GWT
- **Familiarity to JS developers**
  - Lowest: GWT
- **Traditional Ajax support**
  - Tie
- **Server communication**
  - Leader: GWT
  - 2nd tier: DWR, JSON-RPC
- **Usage in industry**
  - Leader: jQuery
  - 2nd tier: Ext-JS, Dojo, YUI, Prototype, Scriptaculous, GWT
- **Looking ahead**
  - All these entries are likely to change significantly
  - Lurking on the horizon: Google "Closure" library

Books and References

- ***Dojo: The Definitive Guide***
  - by Matthew A. Russell
- ***Mastering Dojo***
  - by Craig Riecke et al
- ***Concise Guide to Dojo***
  - by Leslie M. Orchard
- **http://dojotoolkit.org/docs**
  - Moderately complete
  - Moderately well organized
  - Large number of explicit examples

Summary

- **Assigning event handlers programmatically**

```
dojo.addOnLoad(function() {
  dojo.query("#some-id").onclick(someFunction);
});
```

- **General Ajax requests (data-centric Ajax)**

```
dojo.xhrGet({ url: "relative-address",
              load: handlerFunction,
              form: "form-id,
              handleAs: "json" });
```

- **Inserting HTML (content-centric Ajax)**

```
dojo.require("dojo.NodeList-html");
dojo.xhrGet({ … (omit handleAs) … });
function handlerFunction(text) {
  dojo.query("#some-id").html(text);
}
```


Questions?