



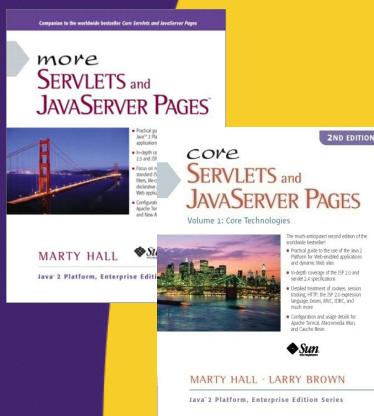
The jQuery JavaScript Library

Part I: Ajax Support

(jQuery 1.4 Version)

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

Customized Java EE Training: <http://courses.coreservlets.com/>
Servlets, JSP, JSF 2.0, Struts, Ajax, GWT 2.0, Spring, Hibernate, SOAP & RESTful Web Services, Java 6.
Developed and taught by well-known author and developer. At public venues or onsite at *your* location.



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, servlets/JSP (intermediate and advanced), Struts, JSF 1.x, JSF 2.0, Ajax, GWT 2.0 (with GXT), 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, Web Services, Ruby/Rails

Contact hall@coreservlets.com for details

Topics in This Section

- Overview of jQuery
- Installation and documentation
- Quick summary of jQuery selectors
- Data centric Ajax: the “\$.ajax” function
 - Basics
 - Options
 - Sending data: static String, data object, String from the “serialize” function
- Content-centric Ajax: the “load” function
- Handling JSON data
- Comparing Ajax support to other libraries
 - Prototype, Dojo, Ext-JS, GWT, YUI, Google Closure

5

© 2010 Marty Hall



Introduction

Customized Java EE Training: <http://courses.coreservlets.com/>

Servlets, JSP, JSF 2.0, Struts, Ajax, GWT 2.0, Spring, Hibernate, SOAP & RESTful Web Services, Java 6.

Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

Overview of jQuery

- **Ajax utilities (this tutorial)**
 - General: `$.ajax(...)`, `$(...).load(...)`
 - Shortcuts: `$.get`, `$.post`, `$.getJSON`
- **DOM search and manipulation utilities**
 - `$("p.myStyle").addClass("extraStyle").show();`
- **Simple animation**
 - Not as extensive as Scriptaculous, but easy to use
- **Cross-browser event model**
 - Assigns handlers programmatically, hides browser differences
- **General JavaScript utilities**
 - Functions operating on strings and arrays
- **Rich GUIs**
 - jQuery UI provides widgets, fancier effects, drag/drop

7

Ajax Utilities

- **`$.ajax({options})`**
 - Makes an Ajax request. Example:
 - `$.ajax({ url: "address", success: responseHandler});`
 - The response handler is passed the response text, not the response object. Don't forget the "." before "ajax"!
- **`load(url)`**
 - Makes Ajax request and loads result into HTML element
 - `$("#some-id").load("address");`
 - A data string or object is an optional second arg
- **Shortcuts**
 - `$.get`, `$.post`, `$.getJSON`
 - Slightly simpler forms of `$.ajax`. However, they support fewer options, so many developers just use `$.ajax`.

8

Downloading and Installation

- **Download**

- <http://jquery.com/>
 - For development, use uncompressed file
 - E.g., jquery-1.4.3.js
 - For deployment, use compressed file
 - E.g., jquery-1.4.3.min.js
- For easier upgrades without changing HTML files
 - Rename file to jquery.js (or possibly jquery-1.4.js)

- **Installation**

- Load jquery.js before any JS files that use jQuery
 - That's it: just a single file to load for core jQuery.
 - jQuery UI (covered in later tutorial) is a bit more involved

- **Online API and tutorials**

- <http://docs.jquery.com/>

9

Browser Compatibility

- **Firefox**

- 2.0 or later (vs. 1.5 or later for Prototype)

- **Internet Explorer**

- 6.0 or later (does not work in IE 5.5)

- **Chrome**

- All (i.e., 1.0 or later)

- **Safari**

- 3.0 or later (vs. 2.0 or later for Prototype)

- **Opera**

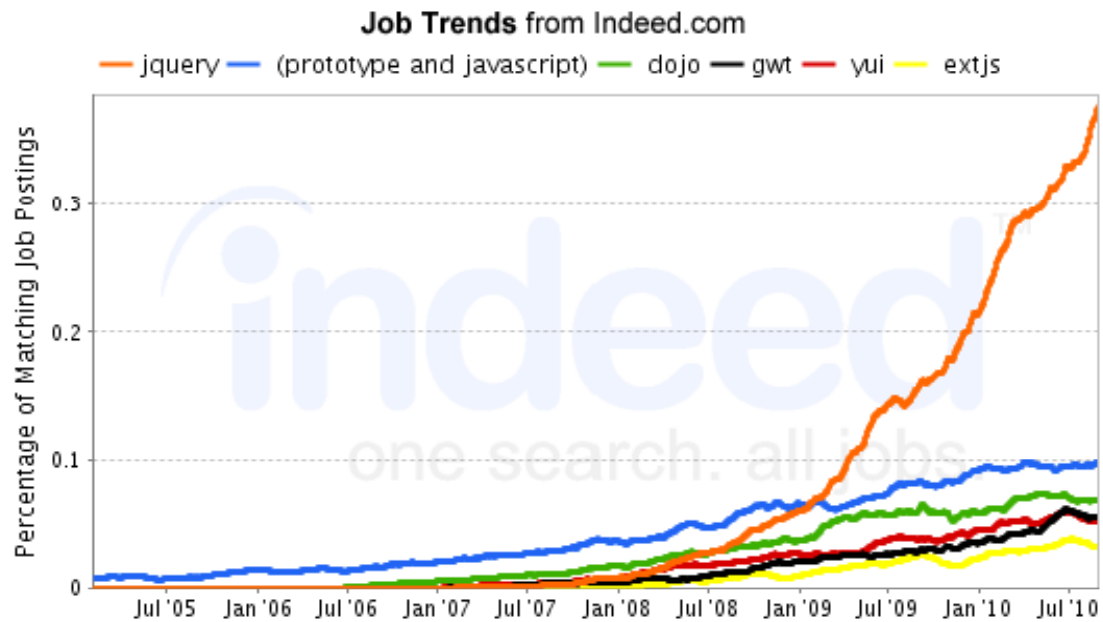
- 9.0 or later (vs. 9.25 or later for Prototype)

- **To check**

- Run the test suite at <http://jquery.com/test/>

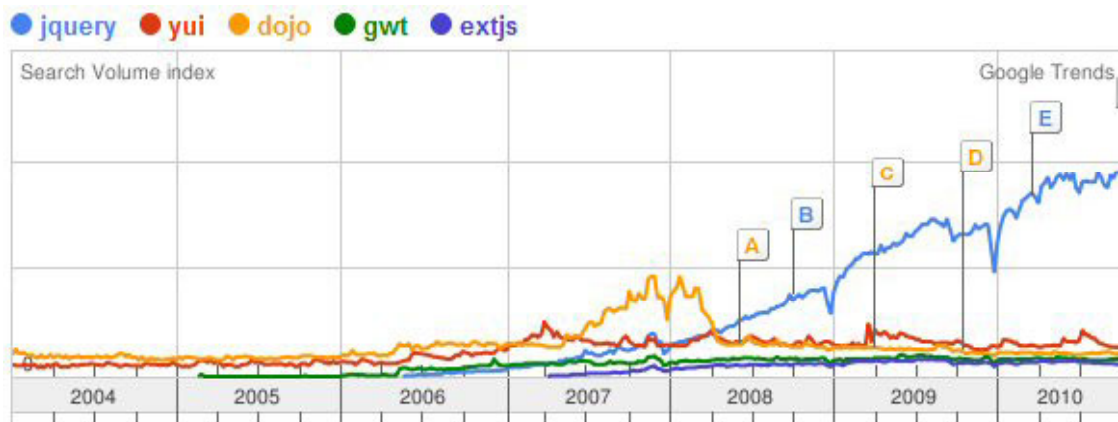
10

Industry Usage (Job Postings)



11

Industry Usage (Google Searches)



12



jQuery Selectors: Basics

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

Customized Java EE Training: <http://courses.coreservlets.com/>
Servlets, JSP, JSF 2.0, Struts, Ajax, GWT 2.0, Spring, Hibernate, SOAP & RESTful Web Services, Java 6.
Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

Selecting DOM Elements

- **Idea**
 - Use `$("css selector")` to get a set of DOM elements
 - Then, perform operations on each (see next page)
 - Much more detail given in next tutorial section
- **Examples**
 - `$("#some-id")`
 - Return 1-element set (or empty set) of element with id
 - Simplest use, and most common for Ajax (note the "#")
 - `$("p")`
 - Return all p elements
 - `$(".blah")`
 - Return all elements that have class="blah"
 - `$("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**
 - `$("#some-id").val()`
 - Returns value of input element. Used on 1-element sets.
 - `$("#selector").each(function)`
 - Calls function on each element. “this” set to element.
 - `$("#selector").addClass("name")`
 - Adds CSS class name to each. Also `removeClass`, `toggleClass`
 - `$("#selector").hide()`
 - Makes invisible (display: none). Also `show`, `fadeOut`, `fadeIn`, etc.
 - `$("#selector").click(function)`
 - Adds onclick handler. Also `change`, `focus`, `mouseover`, etc.
 - `$("#selector").html("<tag>some html</tag>")`
 - Sets the innerHTML of each element. Also `append`, `prepend`
- **Chaining**
 - `$("#a").click(func1).addClass("name").each(func2)`

15

Example: Randomizing Background Colors (JavaScript)

```
function randomizeHeadings() {  
    $("#h3").each(setRandomStyle);  
    $("#h3.green").hide("slow");  
}  
  
function setRandomStyle() {  
    $(this).addClass(randomStyle());  
}  
  
function randomStyle() {  
    var styles = ["red", "yellow", "green"];  
    return(randomElement(styles));  
}  
  
function randomElement(array) {  
    var index = Math.floor(Math.random()*array.length);  
    return(array[index]);  
}
```

Call setRandomStyle function on each h3 element

Slowly hide every h3 that has CSS style "green"

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

16

Example: Randomizing Colors (JavaScript Continued)

```
function revertHeadings() {  
    $("h3.green").show("slow");  
    $("h3").removeClass("red").removeClass("yellow")  
        .removeClass("green");  
}  
  
$(function() {  
    $("#button1").click(randomizeHeadings);  
    $("#button2").click(revertHeadings);  
});
```

Like smart window.onload. Explained in next section.

Sets onclick handlers

17

Example: Randomizing Colors (Style Sheet)

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

...

Names set by setRandomStyles function

18

Example: Randomizing Colors (HTML)

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

During development, renamed jquery-1.4.3.js to jquery.js and used it here. For deployment, replaced it with renamed jquery-1.4.3.min.js

19

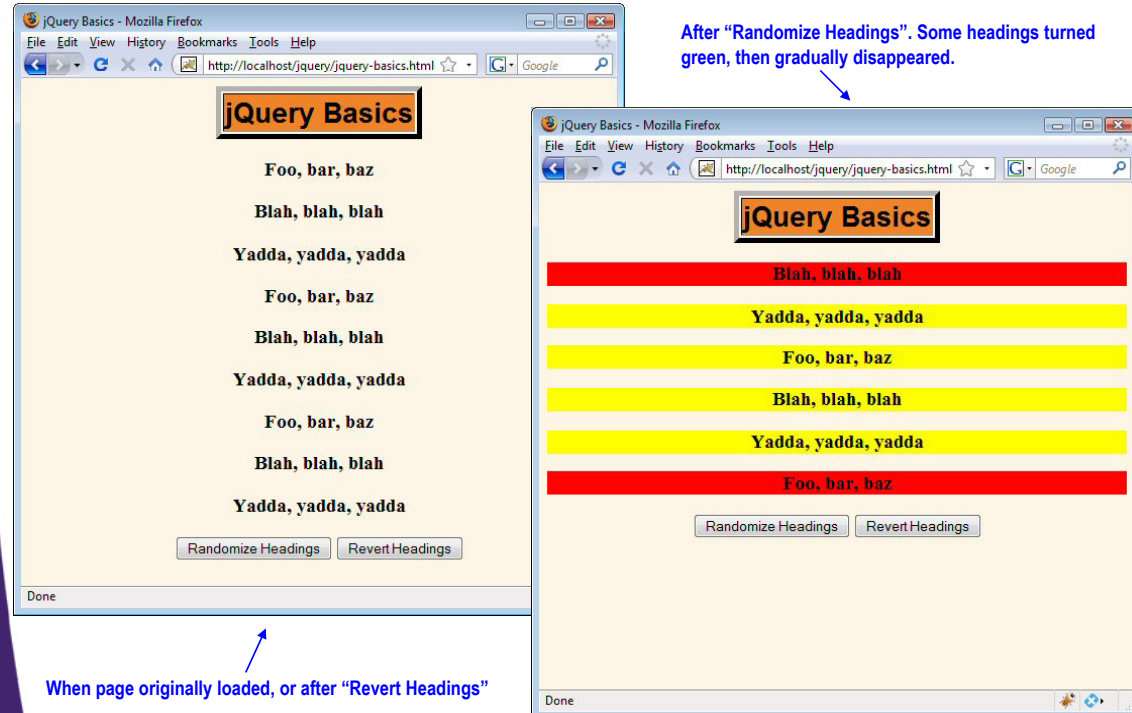
Example: Randomizing Colors (HTML Continued)

```
...
<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>
<input type="button" id="button1"
        value="Randomize Headings"/>
<input type="button" id="button2"
        value="Revert Headings"/>
...
```

The ids to which click handlers are attached.

20

Example: Randomizing Colors (Results)



21

Understanding Operations on Sets of Elements

- **Instead of this**

```
function randomizeHeadings() {  
    $("h3").each(setRandomStyle);  
    $("h3.green").hide("slow");  
}  
function setRandomStyle() {  
    $(this).addClass(randomStyle());  
}
```
- **Why can't I simply do this?**

```
function randomizeHeadings() {  
    $("h3").addClass(randomStyle())  
    $("h3.green").hide("slow");  
}
```

22



\$.ajax: Basics

Customized Java EE Training: <http://courses.coreservlets.com/>
Servlets, JSP, JSF 2.0, Struts, Ajax, GWT 2.0, Spring, Hibernate, SOAP & RESTful Web Services, Java 6.
Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

\$.ajax: Basic Syntax

- **\$.ajax(optionsObject)**
 - Minimal form: `$.ajax({url: "address", success: funct});`
 - Don't forget the ".". It is `$.ajax(...)`, not `$ajax(...)`.
 - The handler function gets the response text, not the response object. jQuery guesses if it should be plain text, XML, or JSON from the response type. If you want to enforce that handler gets given type, use `dataType` option
- **Options for \$.ajax({...})**
 - Almost-always used
 - url, success
 - Other common options
 - cache, data, dataType, error, type, username, password

More on “success” Handler

- **Simplest form**

- function someHandler(text) { ... }
- Note that it gets the response *text*, not the response object
- And, “text” can really be XML or JSON, depending on the dataType option

- **Full form**

- function someHandler(text, status, request) { ... }
- text
 - Response data from server
- status
 - String describing the status: "success" or "notmodified". Rarely useful. In error handlers, the status string is more meaningful.
- request
 - The raw XMLHttpRequest object.

25

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() { someFunc(request); };
    request.open("GET", "some-url", true);
    request.send(null);
}
```

26

Data-Centric Ajax with and without Toolkits

- **jQuery (handler passed response text)**
`$.ajax({url: "address",
 success: handlerFunc});`
- **Prototype (handler passed response object)**
`new Ajax.Request("address",
 {onSuccess: handlerFunc});`
- **Ext (handler passed response object)**
`Ext.Ajax.request({url: "address",
 success: handlerFunc});`
- **Dojo (handler passed response text)**
`dojo.xhrGet({url: "address",
 load: handlerFunc});`

27

\$.ajax Example Code: JavaScript

```
function showTime1() {  
    $.ajax({ url: "show-time.jsp",  
            success: showAlert,  
            cache: false });  
}
```

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

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

This is the response text, not the response object. Use three-argument version if you need the raw XMLHttpRequest object. Also note that recent Firefox versions do not let you pass native functions here, so you cannot use alert instead of showAlert for the success property.

28

\$.ajax Example Code: HTML

```
...
<head><title>jQuery and Ajax</title>...
<script src="./scripts/jquery.js"
        type="text/javascript"></script>
<script src="./scripts/jquery-ajax.js"
        type="text/javascript"></script>
</head>
<body>...
<fieldset>
<legend>$.ajax: Basics (Using onclick
        handler in HTML)</legend>
    <input type="button" value="Show Time"
           onclick='showTime1()' />
</fieldset>
```

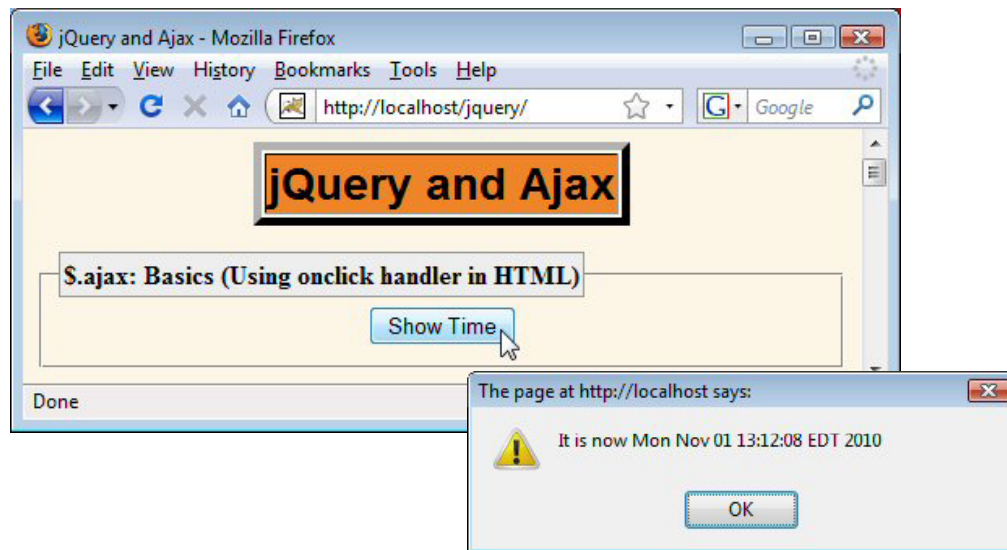
29

\$.ajax Example Code: JSP

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

30

\$.ajax: Results



31

Registering Event Handlers in JavaScript

- **Basic approach**
 - Previous example set the onclick handler in the HTML. Although this is common with some Ajax libraries, jQuery advocates setting it in the JavaScript instead
 - Often referred to as “unobtrusive JavaScript”: no explicit JavaScript anywhere in the HTML page
- **jQuery support**
 - `$(function() {...});`
 - Function runs after the DOM is loaded, but does not wait for images, as with window.onload
 - Use this approach to set up *all* event handlers
 - `$("#some-id").click(someHandler);`
 - Assigns to onclick handler. Handler is passed an Event object with characteristics that are unified across browsers

32

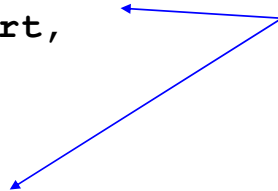
Redoing Time Alert: JavaScript

```
$(function() {  
    $("#time-button-1").click(showTime1);  
});
```

```
function showTime1() {  
    $.ajax({ url: "show-time.jsp",  
            success: showAlert,  
            cache: false });  
}
```

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

These two functions
are unchanged from
previous example.



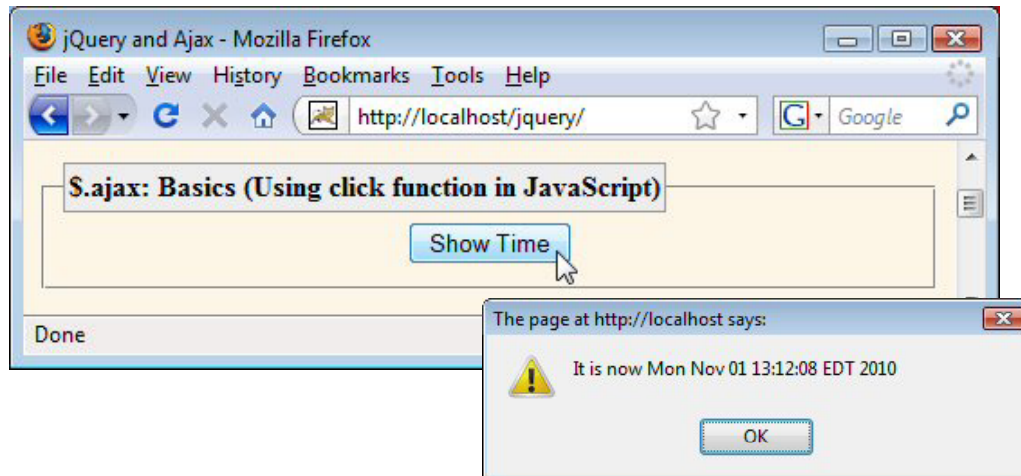
33

Redoing Time Alert: HTML

```
<fieldset>  
<legend>$.ajax: Basics (Using click  
    function in JavaScript)</legend>  
    <input type="button" value="Show Time"  
        id='time-button-1' />  
</fieldset>
```

34

Redoing Time Alert: Results



Works exactly the same as previous example.

35

© 2010 Marty Hall



\$.ajax: Sending Data

Customized Java EE Training: <http://courses.coreservlets.com/>
Servlets, JSP, JSF 2.0, Struts, Ajax, GWT 2.0, Spring, Hibernate, SOAP & RESTful Web Services, Java 6.
Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

Overview

- **`$.ajax({ url: ..., success: ..., data: ... });`**
 - Can be a String, in which case it is sent unchanged.
 - On end of URL or in POST data, depending on HTTP type
 - String can be built automatically using “serialize” function
 - Can be an object, in which case query string gets built out of property names and URL-encoded property values
 - Works identically to “parameters” option in Prototype
- **Equivalent examples**
 - `$.ajax({ ... data: "param1=foo+bar%21¶m2=baz" });`
 - `$.ajax({ ... data: { param1: "foo bar!", param2: "baz" } });`
- **Note: the val function**
 - val reads the value of an input element
 - `var text = $("#some-textfield-id").val();`
 - Works for all input elements, even multiselectable select elements (in which case it returns an array)

37

Three Alternatives

- **Explicit string**
 - Supply an explicit string for data property. We will hardcode it in this example, but it can also be built from data in the page as with earlier Ajax examples.
 - `$.ajax({url: ..., data: "a=foo&b=bar", success: ...});`
- **Data object**
 - Supply an object for data property. Property names become param names and URL-encoded property values become param values. *URL-encoding of values is automatic.*
 - `var params = { a: "value 1", b: "another value!"};`
 - `$.ajax({url: ..., data: params, success: ...});`
- **String built by “serialize”**
 - Give id to the form. Give names (not ids) to input elements. When you call serialize on form, it builds the same query string as a browser would on normal form submission.
 - `$.ajax({url: ..., data: $("#form-id").serialize(), success: ...});`

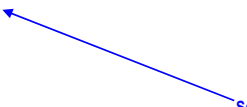
38

The “data” Option with Explicit String: JavaScript

```
$(function() {
    $("#data-button-1").click(showParams1);
    ...
});

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

function showParams1() {
    $.ajax({ url: "show-params.jsp",
            data: "param1=foo&param2=bar",
            success: showAlert });
}
```



Same function used in earlier examples.

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

39

The “data” Option with Explicit String: HTML

```
...
<fieldset>
    <legend>$.ajax: The 'data' Option
        (Fixed String)</legend>
    <input type="button" value="Show Params"
        id="data-button-1"/>
</fieldset>
```

40

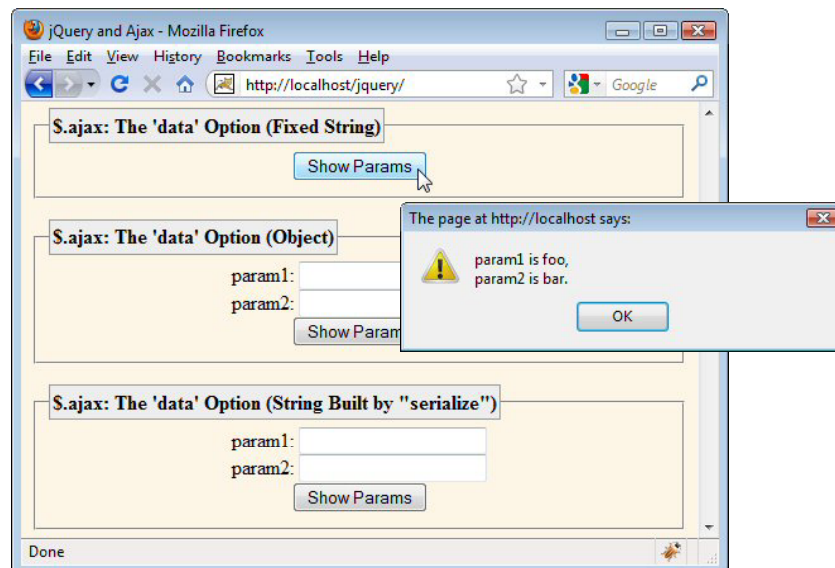
The “data” Option with Explicit String: JSP

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

All three examples with the “data” option use this same JSP page.

41

The “data” Option with Explicit String: Results



42

The “data” Option with Data Object: JavaScript

```
$(function() {  
    $("#data-button-2").click(showParams2);  
    ...  
});  
  
function showAlert(text) {  
    alert(text);  
}  
  
function showParams2() {  
    var params =  
        { param1: $("#field1").val(),  
          param2: $("#field2").val() };  
    $.ajax({ url: "show-params.jsp",  
             data: params,  
             success: showAlert });  
}
```

43

The “data” Option with Data Object: HTML

```
...  
<fieldset>  
    <legend>$.ajax: The 'data' Option  
        (Object)</legend>  
    param1:  
    <input type="text" id="field1"/><br/>  
    param2:  
    <input type="text" id="field2"/><br/>  
    <input type="button" value="Show Params"  
        id="data-button-2"/>  
</fieldset>
```

44

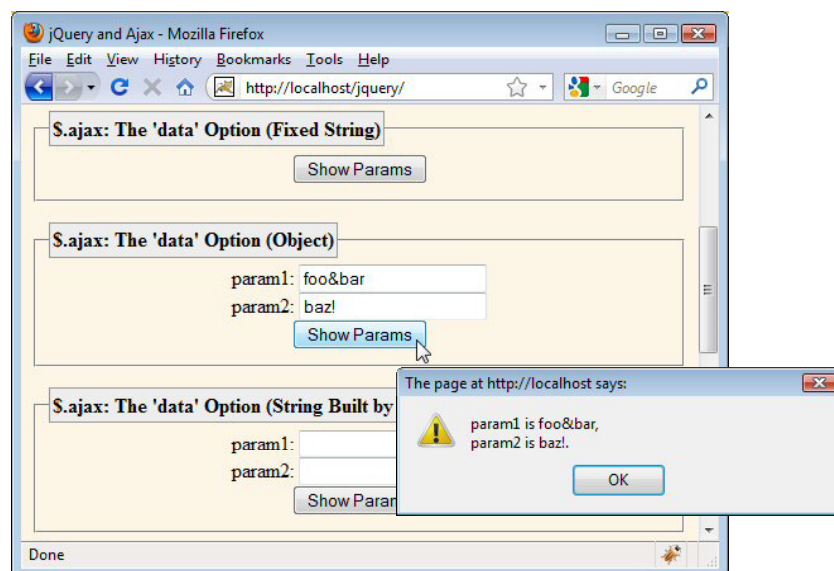
The “data” Option with Data Object: JSP

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

All three examples with the “data” option use this same JSP page.

45

The “data” Option with Data Object: Results



46

The “data” Option with String from “serialize”: JavaScript

```
$(function() {  
    $("#data-button-3").click(showParams3);  
    ...  
});  
  
function showAlert(text) {  
    alert(text);  
}  
  
function showParams3() {  
    $.ajax({ url: "show-params.jsp",  
            data: $("#data-form").serialize(),  
            success: showAlert });  
}
```

47

The “data” Option with String from “serialize”: HTML

```
...  
<fieldset>  
    <legend>$.ajax: The 'data' Option  
        (String Built by "serialize")</legend>  
    <form id="data-form" action="#">  
        param1:  
        <input type="text" name="param1"/><br/>  
        param2:  
        <input type="text" name="param2"/><br/>  
        <input type="button" value="Show Params"  
            id="data-button-3"/>  
    </form>  
</fieldset>
```

48

The reason for action="#" on the form is that, technically, the action attribute of form is required. So, just to satisfy formal HTML validators, we put in dummy action value. But, of course, the action is not used in any way.

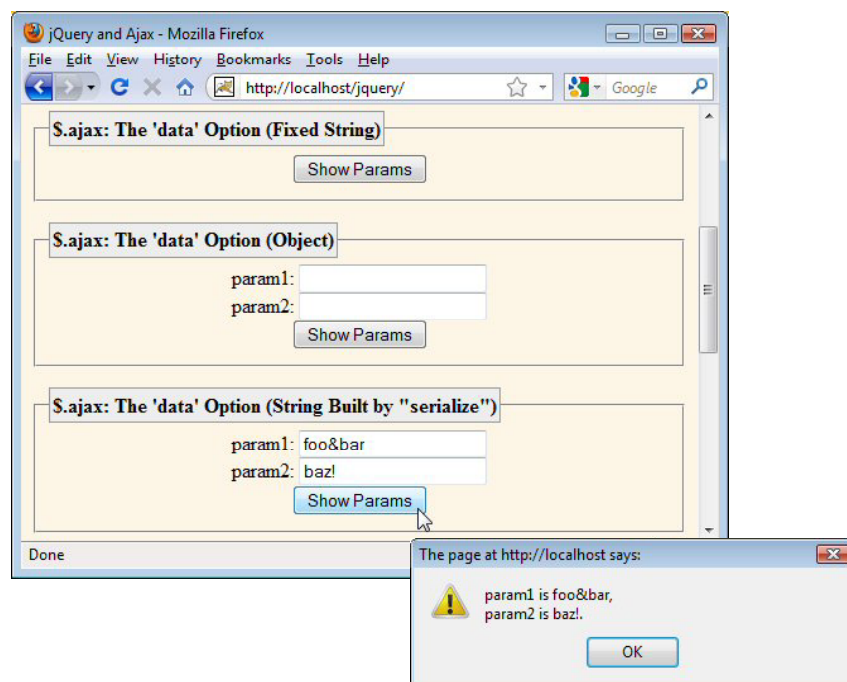
The “data” Option with String from “serialize ”: JSP

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

All three examples with the “data” option use this same JSP page.

49

The “data” Option with String from “serialize ”: Results



50



\$.ajax: Options and Shortcuts

Customized Java EE Training: <http://courses.coreservlets.com/>
Servlets, JSP, JSF 2.0, Struts, Ajax, GWT 2.0, Spring, Hibernate, SOAP & RESTful Web Services, Java 6.
Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

Overview

- **Options (almost) always used: url, success**
 - \$.ajax({url: "some-address", success: someFunction});
 - success is not strictly required; you might want to just fire off some data to the server and not display anything
- **Common options: example**

```
$.ajax({  
    url: "address",  
    success: successHandlerFunction,  
    data: { param1: "foo bar", param2: "baz"},  
    error: errorHandlerFunction,  
    cache: false,  
    dataType: "json",  
    username: "resig",  
    password: "scriptaculous-fan" });
```

Options

Name	Description	Default
async	Should the request be asynchronous? Use synchronous requests with caution since they lock up the browser.	true
beforeSend	Function to modify XMLHttpRequest object before it is sent (e.g., to set custom headers). The XHR is automatically passed to function.	None
cache	Is browser permitted to cache the page? Set to false if you use GET and you could get different responses back from the same data. Equivalent to having the server send Cache-Control: no-cache and Pragma: no-cache.	true (except false if dataType is script or json)
complete	Function to be called after error or success function is finished.	None
contentType	Content-Type of data sent to server. Rarely needed.	application/x-www-form-urlencoded
data	Data to send to server (possibly after conversion). Sent in the appropriate place depending on whether it is GET or POST. Can be a String or an object. If a String, sent unchanged. If an object, property names become param names and property values get URL-encoded and become param values. & and = inserted automatically. If a value is an array, it is serialized with repeated param names.	Empty

53

Options (Continued)

Name	Description	Default
dataFilter	Response-data sanitizing function. Rarely used.	None
dataType	The format in which to pass the response to the handler function. Legal values are text, html (same as text except embedded scripts are run), xml, json, jsonp (JSON with Padding), and script (evaluates the response as JavaScript and returns it as plain text).	html , xml, or json (makes intelligent guess)
error	Function to be called if request fails. Function is passed 3 args: the XHR object, a string describing the error type, and an optional exception object. Possible values for the second argument are null, "timeout", "error", "notmodified" and "parsererror".	None
global	jQuery lets you set global defaults for various handlers: should they be used if set?	true
ifModified	Should the request be considered successful only if the response has changed since the last request (based on the Last-Modified header)?	false
jsonp	Override the callback function name in a jsonp request. JSONP is a JSON extension in which the name of a callback function is specified as an input argument of the call itself.	callback
password username	Username and password to be used in response to HTTP authentication request.	None

54

Options (Continued)

Name	Description	Default
processData	Should the value of the “data” property, if an object, be turned into a URL-encoded query string?	true
scriptCharset	Forces the request to be interpreted as a certain charset. Only for requests with dataType of “jsonp” or “script” and type of “GET”.	None
success	Function to be called if request succeeds. Function passed 3 args: the data returned from the server (formatted according to the dataType property), a string describing the status, and the XHR.	None
timeout	Timeout in milliseconds. If request takes longer, the error handler will be called instead of the success handler.	Global timeout, if set via \$.ajaxSetup
traditional	Should data with arrays be serialized in traditional manner (shallow), or recursively (deep).	false
type	The HTTP method to use for the request. “get” and “post” are main options, but some browsers support other methods.	get
url	The address to request. Should be a relative URL.	None
xhr	Callback for creating your own custom XMLHttpRequest object.	ActiveXObject if available (IE), XMLHttpRequest otherwise

55

Shortcuts for \$.ajax: Equivalent Forms

- **\$.get**
 - \$.get("url", dataObj, someFuncnt)
 - \$.ajax({url: "url", data: dataObj, success: someFuncnt});
- **\$.post**
 - \$.post("url", dataObj, someFuncnt)
 - \$.ajax({url: "url", data: dataObj, success: someFuncnt, type: "post"});
- **\$.getJSON**
 - \$.getJSON("url", dataObj, someFuncnt)
 - \$.ajax({url: "url", data: dataObj, success: someFuncnt, dataType: "json"});
- **Note**
 - get and post take the type as an optional fourth argument

56

Pros and Cons of Shortcuts

- **Advantages of shorthand functions**
 - Slightly shorter and (arguably) clearer
- **Disadvantages of shorthand functions**
 - If you want additional options later, you have to change existing code more drastically
 - There is no direct argument for supplying error handler. Instead, you must call `ajaxError` to set up error handler.
 - If you don't have data, you have to pass in null. This is less convenient than just omitting the "data" property.
 - `$.get("url", null, someHandler);` vs.
 - `$.ajax({url: "url", success: someHandler});`
 - In older jQuery versions, couldn't use `serialize`.
 - In jQuery 1.3 and earlier, the data cannot be a string. This means you cannot use the `serialize` function to automatically build the query string from the form

57

© 2010 Marty Hall



Simplifying Inserting Results into HTML: the "load" Function

Customized Java EE Training: <http://courses.coreservlets.com/>

Servlets, JSP, JSF 2.0, Struts, Ajax, GWT 2.0, Spring, Hibernate, SOAP & RESTful Web Services, Java 6.

Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

load: Basic Syntax

- **Basic forms**

- `$("#result-area-id").load("url");`
- `$("#result-area-id").load("url", data);`
- `$("#result-area-id").load("url", data, handlerFunction);`

- **Helper utilities**

- **val**: reading the value of an input element
 - `var text = $("#some-textfield-id").val();`
 - Works for all input elements, even multiselectable select elements (in which case it returns an array)
- Filtering returned HTML
 - Add jQuery selectors to the URL. Only elements that match will be inserted. E.g., to insert only li elements:
 - `$("#result-id").load("address li");`
 - Go very light with filtering: use data-centric Ajax instead.

59

Content-Centric Ajax with and without Toolkits (Basic JS Only)

```
function getRequestObject() { ... }

function ajaxResult(address, resultRegion) {
    var request = getRequestObject();
    request.onreadystatechange =
        function() { showResponseText(request,
                                     resultRegion); };
    request.open("GET", address, true);
    request.send(null);
}

function showResponseText(request, resultRegion) {
    if ((request.readyState == 4) &&
        (request.status == 200)) {
        document.getElementById(resultRegion).innerHTML =
            request.responseText;
    }
}
```

60

Content-Centric Ajax with and without Toolkits (\$.ajax)

```
function ajaxResult(address, resultRegion) {
    $.ajax({
        url: address,
        success: function(text) {
            showResponseText(text, resultRegion);
        }
    });
}

function showResponseText(text, resultRegion) {
    $(resultRegion).html(text);
}
```

Note: use the "html" function (as above) to replace the innerHTML of the element. Use "append" to add to the end of the innerHTML of the element.

61

Content-Centric Ajax with and without Toolkits (Libraries)

- **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*

- **Ext-JS**

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

62

load Example: JavaScript

```
$(function() {  
    $("#load-button").click(insertParams);  
    ...  
});  
  
function insertParams() {  
    $("#params-result").load("show-params.jsp",  
        $("#load-form").serialize());  
}
```

63

load Example: HTML

```
...  
<fieldset>  
    <legend>$.load: Simplifying HTML Insertion</legend>  
    <form id="load-form" action="#">  
        param1:  
        <input type="text" name="param1"/>  
        <br/>  
        param2:  
        <input type="text" name="param2"/>  
        <br/>  
        <input type="button" value="Show Params"  
            id="params-button"/>  
        <h2 id="params-result"></h2>  
    </form>  
</fieldset>
```

64

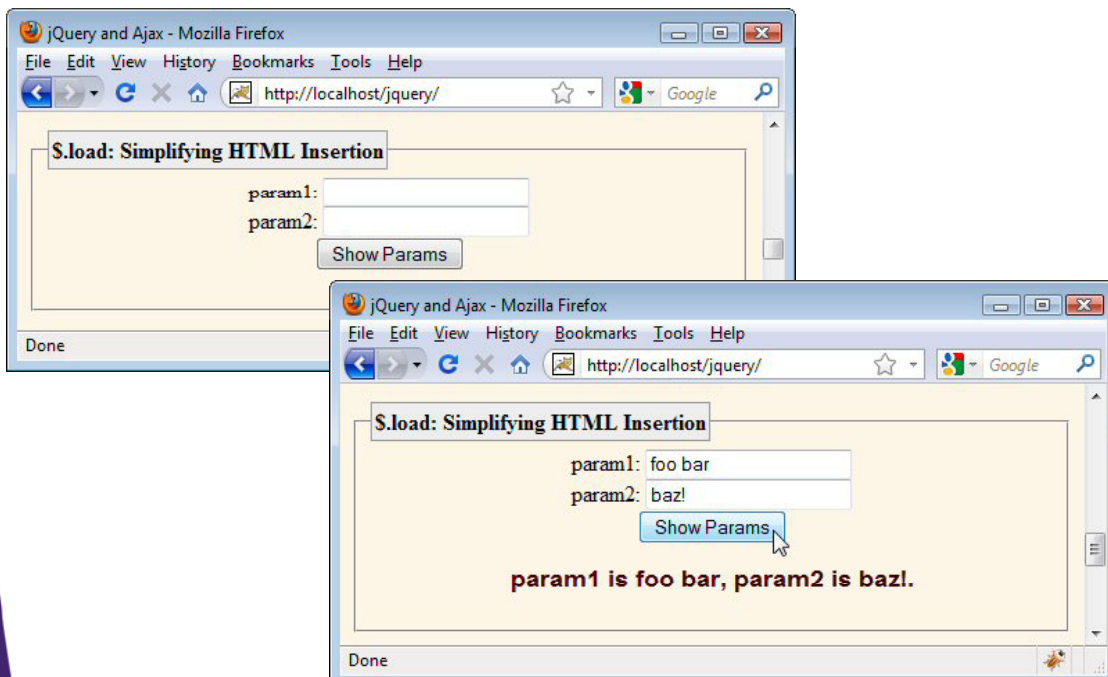
load Example: JSP

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

Unchanged from previous examples

65

load Example: Results



66



Handling JSON Data

Customized Java EE Training: <http://courses.coreservlets.com/>

Servlets, JSP, JSF 2.0, Struts, Ajax, GWT 2.0, Spring, Hibernate, SOAP & RESTful Web Services, Java 6.

Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

Approach

- **Server**
 - Returns JSON object with no extra parens. E.g.:
 - `{ "cto": "Resig ", "ceo": "Gates ", "coo": "Ellison" }`
- **Code that calls \$.ajax**
 - Specifies dataType of json. E.g.:
 - `$.ajax({ url: address, success: handler, dataType: "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) {
 $("#some-id").html("Chief Technology Officer is " +
 companyExecutives.cto + "");
}`

Strict JSON

- **JSON in practice**
 - The way you would type a data structure into JavaScript.
 - This is what “eval”, Prototype, and jQuery 1.3 support
- **Strict JSON according to json.org**
 - Subset of JavaScript where
 - Object property names must be in double quotes
 - Strings are represented with double quotes only (not single quotes)
 - This is what jQuery 1.4 supports. Since this is what is clearly described at json.org, you should follow this format when sending JSON from the server.
- **MIME type for JSON from server**
 - RFC 4627 says JSON should have "application/json" type
 - No known libraries enforce this

69

JSON Example Code: Core JavaScript

```
$(function() { ...
    $("#nums-button").click(showNums);
});

function showNums() {
    $.ajax({ url: "show-nums",
            dataType: "json",
            success: showNumberList });
}

function showNumberList(jsonData) {
    var list = makeList(jsonData.fg, jsonData.bg,
                        jsonData.fontSize,
                        jsonData.numbers);
    $("#nums-result").html(list);
}
```

```
graph TD
    Strings --> fg
    Strings --> bg
    int --> numbers
    ArrayOfDoubles --> fontSize
```

70

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");
}
```

71

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>");
}
```

72

JSON Example Code: HTML

```
<fieldset>
<legend>$.ajax: Treating Response as JSON</legend>
  <input type="button" value="Show Nums"
        id='nums-button' />
  <div id="nums-result3"></div>
</fieldset>
```

73

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);
    }
}
```

74

JSON Example Code: JSP

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

- **Notes**

- Quotes around property names. Double, not single, quotes
- Client-side code does not need wrap in parens and pass to “eval”. JSON evaluation handled automatically by jQuery
- Types
 - fg and bg: Strings
 - fontSize: int
 - numbers: Array of doubles

75

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"  
    };  
  
    /** One of the official HTML color names, at random. */  
  
    public static String randomColor() {  
        return(RandomUtils.randomElement(colors));  
    }  
  
    private ColorUtils() {} // Uninstantiatable class  
}
```

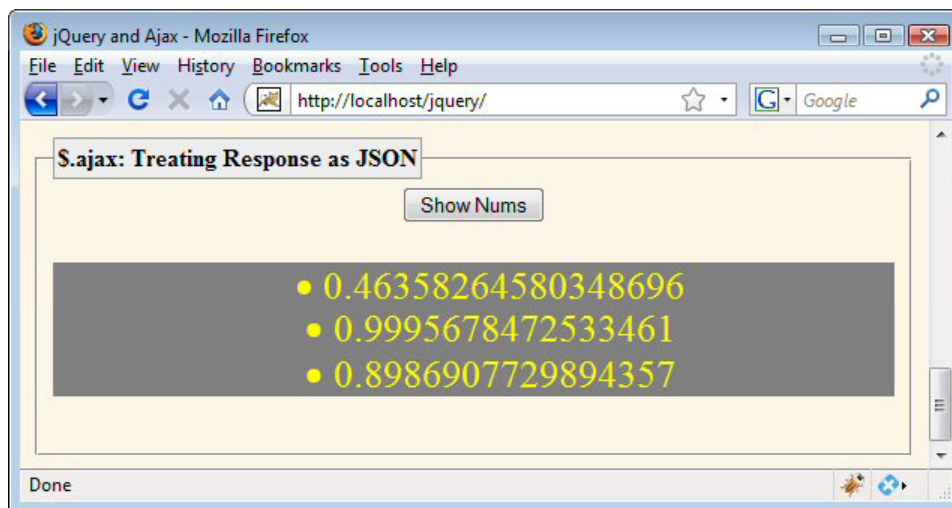
76

JSON Example Code: Auxiliary Java Code

```
public class RandomUtils {  
    private static Random r = new Random();  
  
    public static int randomInt(int range) {  
        return(r.nextInt(range));  
    }  
  
    public static int randomIndex(Object[] array) {  
        return(randomInt(array.length));  
    }  
  
    public static <T> T randomElement(T[] array) {  
        return(array[randomIndex(array)]);  
    }  
}
```

77

JSON Example: Results



78



Wrap-up

Customized Java EE Training: <http://courses.coreservlets.com/>

Servlets, JSP, JSF 2.0, Struts, Ajax, GWT 2.0, Spring, Hibernate, SOAP & RESTful Web Services, Java 6.
Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

“Best” JavaScript Libraries

- **General JS programming**
 - Leaders: Closure, Prototype
- **Other programming (Java)**
 - Leader (only): GWT
- **DOM manipulation**
 - Leader: jQuery
 - Others are 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, Google Closure
- **Familiarity to JS developers**
 - Lowest: GWT
 - 2nd-lowest: Google Closure
- **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
 - Lowest: Google Closure
- **Support for extra-large projects**
 - Leaders: Closure, GWT
 - 2nd tier: Ext-JS, YUI, Dojo

Books and References

- ***jQuery in Action***
 - by Bear Bibeault, Yehuda Katz, and John Resig
 - Makes global variable blunder when showing normal Ajax
- ***jQuery 1.4 Reference Guide***
 - by Jonathan Chaffer and Karl Swedberg
- **<http://docs.jquery.com/>**
 - Very complete
 - Moderately well organized
 - Moderate number of explicit examples
- ***jQuery UI 1.7***
 - By Dan Wellman
 - Looks good, but jQuery UI is changing rapidly (current version is 1.8), so the online docs are perhaps better

81

Summary

- **Assigning event handlers programmatically**

```
$(function() {  
    $("#some-id").click(someFunction);  
});
```
- **General Ajax requests (data-centric Ajax)**

```
$.ajax({ url: "relative-address",  
        success: handlerFunction,  
        data: $("#form-id").serialize(),  
        dataType: "json" });
```
- **Loading result (content-centric Ajax)**

```
$("#result-id").load("relative-address",  
                    $("#form-id").serialize());
```

82



Questions?

Customized Java EE Training: <http://courses.coreservlets.com/>

Servlets, JSP, JSF 2.0, Struts, Ajax, GWT 2.0, Spring, Hibernate, SOAP & RESTful Web Services, Java 6.

Developed and taught by well-known author and developer. At public venues or onsite at *your* location.