Wintellect
Know how.

jQuery AJAX

Jeff Prosise
jeffpro@wintellect.com

Wintellect NOW Consulting/Training Wintellect Know how.

1

AJAX

- AJAX = Asynchronous JavaScript and XML
  - Term coined by Jesse James Garrett in February 2005
  - Despite name, often does NOT use XML
- Uses XMLHttpRequest to send HTTP request to server and retrieve response
  - Round-trips to server without refreshing page
  - Introduced in 1999 with Internet Explorer 5
  - Later adopted by Firefox, Safari, and others
- jQuery simplifies AJAX and works cross-browser

Wintellect NOW Wintellect Know how.

2

AJAX without jQuery

```
var xhr;

if (window.XMLHttpRequest) {
    xhr = new XMLHttpRequest(); // Modern browsers
}
else { // Older browsers
    try {
        xhr = new ActiveXObject("Msxml2.XMLHTTP");
    }
    catch (ex) {
        xhr = new ActiveXObject("Microsoft.XMLHTTP");
    }
}
```

Wintellect NOW Wintellect Know how.

3

AJAX without jQuery, Cont.

```
xhr.open ("GET", "services/getarticle/1003");

xhr.onreadystatechange = function() {
    if (xhr.readyState == 4 && xhr.status >= 200 &&
        xhr.status < 300) {
        var news = document.getElementById("article");
        news.innerHtml = xhr.responseText;
    }
}

xhr.send ();
```

Wintellect NOW Wintellect Know how.

4

AJAX with jQuery

```
$("#article").load("services/getarticle/1003");
```

Wintellect NOW Wintellect Know how.

5

DEMO

The .load() Method

Wintellect NOW Consulting/Training Wintellect Know how.

6

AJAX Shortcut Methods

- Provided to simplify tasks such as fetching and displaying HTML, fetching JSON from a service, and posting forms to a server

Method	Description
.load()	Loads HTML from a server and displays it in a DOM element
jQuery.getJSON()	Loads JSON from a server, parses it, and returns the result
jQuery.getScript()	Loads a JavaScript file from the server and executes it
jQuery.get()	Submits an AJAX requesting containing an HTTP GET command and optionally parses JSON or XML content
jQuery.post()	Submits an AJAX requesting containing an HTTP POST command

7

Using jQuery.getScript()

```
// Load and execute a script file named test.js
$.getScript("scripts/test.js");

// Load and execute test.js, and read the return value
$.getScript("scripts/test.js", function(data) {
    var return = data;
});
```

8

Using jQuery.getJSON()

```
// Fetch a JSON-encoded object with properties named
// firstName and lastName, then display them to the user
$.getJSON("services/getperson/2001", function (result) {
    alert(result.firstName + " " + result.lastName);
});
```

```
{"firstName":"Jeff","lastName":"Prosise"}
```

9

Populating a Drop-Down List

```
// Fetch a JSON-encoded string array from the server and use
// it to populate a drop-down list whose ID is "dropdownlist"
$.getJSON("services/getitems", function (result) {
    var items;
    $(result).each(function () {
        items += "<option>" + this + "</option>";
    });
    $("#dropdownlist").append(items);
});
```

```
["Red","White","Blue"]
```

10

DEMO

jQuery.getJSON()

11

Using jQuery.get() to Fetch JSON

```
// Fetch a JSON-encoded string array from the server and use
// it to populate a drop-down list (equivalent to $.getJSON())
$.get("services/getitems", function (result) {
    var items;
    $(result).each(function () {
        items += "<option>" + this + "</option>";
    });
    $("#dropdownlist").append(items);
}, "json"); // Optional
```

```
["Red","White","Blue"]
```

12

Using jQuery.get() to Fetch XML

```
// Fetch XML data from the server and use it to populate a
// drop-down list
$.get("services/getitems", function (result) {
    var items;
    $(result).each(function () {
        items += "<option>" + this + "</option>";
    });
    $("#dropdownlist").append(items);
}, "xml");
```

```
<Data><Item>Red</Item><Item>White</Item>Blue<Item></Item></Data>
```

13

Using jQuery.post()

```
<form id="loginform">
  <input type="text" id="username" name="username" />
  <input type="password" id="password" name="password" />
  <input type="submit" id="submit" value="Submit" />
</form>

$("#submit").click(function (e) {
    var data = "username=" + $("#username").val() +
        "&password=" + $("#password").val();
    $.post("services/login", data);
    e.preventDefault();
});
```

14

Using .serialize()

```
<form id="loginform">
  <input type="text" id="username" name="username" />
  <input type="password" id="password" name="password" />
  <input type="submit" id="submit" value="Submit" />
</form>

$("#submit").click(function (e) {
    var data = $("#loginform").serialize();
    $.post("services/login", data);
    e.preventDefault();
});
```

15

Getting a Response

```
<form id="loginform">
  <input type="text" id="username" name="username" />
  <input type="password" id="password" name="password" />
  <input type="submit" id="submit" value="Submit" />
</form>

$("#submit").click(function (e) {
    var data = $("#loginform").serialize();
    $.post("services/login", data, function(result) {
        alert(result ? "Login succeeded" : "Login failed");
    }, "json"); // Assumes JSON response containing true or false
    e.preventDefault();
});
```

16

DEMO

jQuery.post()

17

jQuery.ajax()

- General-purpose jQuery AJAX method
  - Used internally by other AJAX methods
  - Often called directly
- Exposes numerous options via options object
  - URL targeted in call
  - Success, error, and completion callbacks
  - HTTP command type and much more
- The heart of jQuery's AJAX support

http://api.jquery.com/jQuery.ajax/

18

Common jQuery.ajax() Options

Option	Description
cache	Set to false to prevent content fetched with GET from being cached
complete	Completion callback function
data	Data to include in the request
error	Error callback function
dataType	Type of data included in the request
ifModified	Set to true to request content only if it has been modified since the last time it was requested (look for "notmodified" completion status)
success	Success callback function
timeout	Time-out interval in milliseconds (default == 0)
type	HTTP method (default == "GET")
url	URL to which the request will be transmitted

19

Fetching HTML Content

```
$.ajax({
    url: "services/getarticle/1003",
    success: function(result) {
        $("#article").html(result);
    }
});
```

20

Guaranteeing Fresh Content

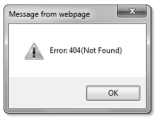
```
$.ajax({
    url: "services/getarticle/1003",
    success: function(result) {
        $("#article").html(result);
    },
    cache: false
});
```

Appends _= query-string parameter to URL to prevent server from returning cached content

21

Detecting Errors

```
$.ajax({
    url: "services/getarticle/1003",
    success: function(result) {
        $("#article").html(result);
    },
    error: function (xhr, status, error) {
        if (status == "error") {
            alert("Error: " + xhr.status + " (" + error + ")");
        }
    }
});
```



22

Applying Time-Outs

```
$.ajax({
    url: "services/getarticle/1003",
    success: function(result) {
        $("#article").html(result);
    },
    error: function (xhr, status, error) {
        if (status == "timeout") {
            alert("Request timed out");
        }
        else if (status == "error") {
            alert("Error: " + xhr.status + " (" + error + ")");
        }
    },
    timeout: 5000 // 5 seconds
});
```

23

jQuery.ajaxSetup()

- Sets defaults for all AJAX calls
  - Options can still be overridden in individual calls
- Related jQuery.ajaxPrefilter() method permits options to be modified on the fly
  - For example, change default time-out for selected calls
- Latest guidance "strongly suggests" that jQuery.ajaxSetup() not be used because it breaks many plugins

http://api.jquery.com/jQuery.ajaxSetup/

24

Using jQuery.ajaxSetup()

```
// Apply a default error handler and time-out interval to all
// subsequent AJAX requests (.load(), $.get(), $.ajax(), etc.)
$.ajaxSetup({
    error: function (xhr, status, error) {
        if (status == "timeout") {
            alert("Request timed out");
        }
        else if (status == "error") {
            alert("Error: " + xhr.status + " (" + error + ")");
        }
    },
    timeout: 5000
});
```

25

DEMO

jQuery.ajax()

Consulting/Training

26

Global AJAX Event Handlers

- Methods that register callbacks for global AJAX events

Method	Description
.ajaxComplete()	Registers a handler to be called when any AJAX request completes
.ajaxError()	Registers a handler to be called when any AJAX request fails
.ajaxSend()	Registers a handler to be called when any AJAX request begins
.ajaxStart()	Registers a handler to be called when the first AJAX request begins
.ajaxStop()	Registers a handler to be called when the last AJAX request completes
.ajaxSuccess()	Registers a handler to be called when any AJAX request completes successfully

27

Using .ajaxError()

```
// Display an error message if an AJAX request fails
$("#error").ajaxError(function() {
    $(this).text("Error!");
});

// Display an error message containing the offending URL
// if an AJAX request fails
$("#error").ajaxError(function(e, xhr, settings, exception) {
    $(this).text("Error calling " + settings.url);
});
```

28

DEMO

jQuery.ajaxSend() and jQuery.ajaxComplete()

Consulting/Training

29

Promises

- In jQuery 1.5+, AJAX methods return jQuery XHR objects implementing promises
  - Contain .done(), .fail(), .always(). and .then() methods
- Promises simplify syntax and allow multiple callbacks to be bound to a single completion
  - Called when bound, even if operation has completed
- Related jQuery.when() method allows multiple promises to be aggregated into one
  - Executes callbacks when all complete or any fail

30

Using Promises

```
var promise = $.ajax({ url: "services/getarticle/1003" });

promise.done(function(result) {
    // Request completed successfully
});

promise.fail(function(xhr, status, error) {
    // Request failed
});

promise.always(function() {
    // Called regardless of whether request succeeded or failed
});
```

31

Chaining Promises

```
$.ajax({ url: "services/getarticle/1003" }).done(function(result) {
    // Request completed successfully
}).fail(function(xhr, status, error) {
    // Request failed
}).always(function() {
    // Called regardless of whether request succeeded or failed
})
```

32

Using .then()

```
$.ajax({ url: "services/getarticle/1003" }).then(function(result) {
    // Request completed successfully
}, function(xhr, status, error) {
    // Request failed
});
```

33

Using jQuery.when()

```
var promise1 = $.ajax({ url: "services/getarticle/1003" });
var promise2 = $.ajax({ url: "services/getarticle/1004" });

var promise = $.when(promise1, promise2); // Combine the promises

promise.done(function(xhr1, xhr2) {
    // Both requests completed successfully
    var result1 = xhr1[0]; // Get result of first call
    var result2 = xhr2[0]; // Get result of second call
});

promise.fail(function(xhr, status, error) {
    // One of the requests failed (other may be incomplete)
});
```

34

Determining Which Call Failed

```
var promise1 = $.ajax({ url: "services/getarticle/1003" });
var promise2 = $.ajax({ url: "services/getarticle/1004" });

var promise = $.when(promise1, promise2); // Combine the promises
  .
  .
  .
promise.fail(function(xhr, status, error) {
    if (promise1.isRejected()) {
        // Call #1 failed
    }
    else if (promise2.isRejected()) {
        // Call #2 failed
    }
});
```

35

DEMO

Promises

Consulting/Training

36