

AJAX HACKS

Tips & Tools for Creating Responsive Web Sites



O'REILLY®

Bruce W. Perry

Ajax Hacks™

by Bruce Perry

Copyright © 2006 O'Reilly Media, Inc. All rights reserved.
Printed in the United States of America.

Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North,
Sebastopol, CA 95472.

O'Reilly books may be purchased for educational, business, or sales promotional use. Online editions are also available for most titles (*safari.oreilly.com*). For more information, contact our corporate/institutional sales department: (800) 998-9938 or *corporate@oreilly.com*.

Editors: Simon St.Laurent

Production Editor: Mary Anne Weeks Mayo

Copyeditor: Rachel Wheeler

Indexer: John Bickelhaupt

Cover Designer: Hanna Dyer

Interior Designer: David Futato

Illustrators: Robert Romano, Jessamyn
Read, and Lesley Borash

Printing History:

March 2006: First Edition.

Nutshell Handbook, the Nutshell Handbook logo, and the O'Reilly logo are registered trademarks of O'Reilly Media, Inc. The *Hacks* series designations, *Baseball Hacks*, the image of an umpire's indicator, and related trade dress are trademarks of O'Reilly Media, Inc.

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and O'Reilly Media, Inc. was aware of a trademark claim, the designations have been printed in caps or initial caps.

While every precaution has been taken in the preparation of this book, the publisher and author assume no responsibility for errors or omissions, or for damages resulting from the use of the information contained herein.

Small print: The technologies discussed in this publication, the limitations on these technologies that technology and content owners seek to impose, and the laws actually limiting the use of these technologies are constantly changing. Thus, some of the hacks described in this publication may not work, may cause unintended harm to systems on which they are used, or may not be consistent with applicable user agreements. Your use of these hacks is at your own risk, and O'Reilly Media, Inc. disclaims responsibility for any damage or expense resulting from their use. In any event, you should take care that your use of these hacks does not violate any applicable laws, including copyright laws.



This book uses RepKover™, a durable and flexible lay-flat binding.

ISBN: 0-596-10169-4

[M]

[4/06]

This excerpt is protected by copyright law. It is your responsibility to obtain permissions necessary for any proposed use of this material. Please direct your inquiries to permissions@oreilly.com.



HACK

#1

Detect Browser Compatibility with the Request Object

Use JavaScript to set up Microsoft's and the Mozilla-based browsers' different request objects.

Browser compatibility is an important consideration. You have to make sure the “engine” behind Ajax’s server handshake is properly constructed, but you can never predict which browsers your users will favor.

The programming tool that allows Ajax applications to make HTTP requests to a server is an object that you can use from within JavaScript code. In the world of Firefox and Netscape (as well as Safari and Opera), this object is named `XMLHttpRequest`. However, continuing with the tradition established by IE 5.0, recent vintages of Internet Explorer implement the software as an ActiveX object named `Microsoft.XMLHTTP` or `Mxml2.XMLHTTP`.



`Microsoft.XMLHTTP` and `Mxml2.XMLHTTP` refer to different versions of software components that are a part of Microsoft XML Core Services (MSXML). Here’s what our contributing IE expert says on this matter:

“If you use `Microsoft.XMLHTTP`, the `ActiveXObject` wrapper will try to initialize the last known good version of the object that has this program (or “prog”) ID. This object, in theory, could be MSXML 1.0, but almost no one these days has that version because it has been updated via Windows Update, IE 6, or another means. MSXML 1.0 was very short-lived. If you use `MSXML2.XMLHTTP`, that signifies to the wrapper to use at least MSXML 2.0 libraries. Most developers do not need to use a specific version of MSXML, such as `MSXML2.XMLHTTP.4.0` or `MSXML2.XMLHTTP.5.0`.”

Although Microsoft and the engineers on the Mozilla project have chosen to implement this object differently, we will refer to the ActiveX and `XMLHttpRequest` objects simply as “request objects” throughout this book, because they have very similar functionality.

As a first step in using Ajax, you must check if the user’s browser supports either one of the Mozilla-based or ActiveX-related request objects, and then properly initialize the object.

Using a Function for Checking Compatibility

Wrap the compatibility check inside a JavaScript function, then call this function before you make any HTTP requests using the object. For exam-

ple, in Mozilla-based browsers such as Netscape 7.1 and Firefox 1.5 (as well as in Safari 2.0 and Opera 8.5), the request object is available as a property of the top-level `window` object. The reference to this object in JavaScript code is `window.XMLHttpRequest`. The compatibility check for these browser types looks like this:

```
if(window.XMLHttpRequest){
    request = new XMLHttpRequest();
    request.onreadystatechange=handleResponse;
    request.open("GET",theURL,true);
    request.send(null);
}
```

The JavaScript variable `request` is to a top-level variable that will refer to the request object.



As an alternative model, the open-source library *Prototype* uses object-oriented JavaScript to wrap the request object into its own object, as in the object `Ajax.Request` (see Chapter 6).

If the browser supports `XMLHttpRequest`, then:

1. `if(window.XMLHttpRequest)` returns true because the `XMLHttpRequest` is not null or undefined.
2. The object will be instantiated with the new keyword.
3. Its `onreadystatechange` event listener (see the section “XMLHttpRequest” earlier in this chapter) will be defined as a function named `handleResponse()`.
4. The code calls the request object’s `open()` and `send()` methods.

What about Internet Explorer users?



Microsoft Internet Explorer–related blogs mentioned, at the time this book went to publication, that IE 7 would support a native `XMLHttpRequest` object.

In this case, the `window.XMLHttpRequest` object will not exist in the browser object model. Therefore, another branch of the `if` test is necessary in your code:

```
else if (window.ActiveXObject){
    request=new ActiveXObject("Microsoft.XMLHTTP");
    if (! request){
        request=new ActiveXObject("Msxml2.XMLHTTP");
    }
}
```

```
    }
    if(request){
        request.onreadystatechange=handleResponse;
        request.open(reqType,url,true);
        request.send(null);
    }
}
```

This code fragment tests for the existence of the top-level window object `ActiveXObject`, thus signaling the use of Internet Explorer. The code then initializes the request using two of a number of possible ActiveX program IDs (here, `Microsoft.XMLHTTP` and `Mxml2.XMLHTTP`).

You can get even more fine-grained when testing for different versions of the IE request object, such as `Mxml2.XMLHTTP.4.0`. In the vast majority of cases, however, you will not be designing your application based on various versions of the MSXML libraries, so the prior code will suffice.

The code then makes one final check for whether the request object has been properly constructed (`if(request){...}`).

Given three chances, if the request variable is still null or undefined, your browser is really out of luck when it comes to using the request object for Ajax!

Here's an example of an entire compatibility check:

```
/* Wrapper function for constructing a request object.
Parameters:
reqType: The HTTP request type, such as GET or POST.
url: The URL of the server program.
asynch: Whether to send the request asynchronously or not. */

function httpRequest(reqType,url,asynch){
    //Mozilla-based browsers
    if(window.XMLHttpRequest){
        request = new XMLHttpRequest();
    } else if (window.ActiveXObject){
        request=new ActiveXObject("Mxml2.XMLHTTP");
        if (! request){
            request=new ActiveXObject("Microsoft.XMLHTTP");
        }
    }
    //the request could still be null if neither ActiveXObject
    //initialization succeeded
    if(request){
        initReq(reqType,url,asynch);
    } else {
        alert("Your browser does not permit the use of all "+
            "of this application's features!");
    }
}
```

```
    }  
  }  
  /* Initialize a request object that is already constructed */  
  function initReq(reqType,url,bool){  
    /* Specify the function that will handle the HTTP response */  
    request.onreadystatechange=handleResponse;  
    request.open(reqType,url,bool);  
    request.send(null);  
  }  
}
```

“Use the Request Object to POST Data to the Server” [Hack #2] shows how to implement a POST request with XMLHttpRequest.