

Build Ajax-based Web sites with PHP

Do it in style with some of the latest frameworks available today

Skill Level: Intermediate

[Ken Ramirez \(ken.ramirez@axsystechgroup.com\)](mailto:ken.ramirez@axsystechgroup.com)

Founder

Axsys Technology Group

02 Sep 2008

Learn the process of writing Asynchronous JavaScript + XML (Ajax) applications using native JavaScript code and PHP. This article introduces a few different frameworks and application program interfaces (APIs) that reduce the amount of code you need to write to achieve a complete Ajax-based Web application.

PHP has been around for quite a few years. It's commonly used as a server-side scripting language to develop Web-based applications fairly quickly and with good results. In fact, some of the most popular Web-based projects such as PHP-Nuke, osCommerce, and Joomla were all developed in PHP and continue to thrive today.

Ajax has also been around for a while, but it's only recently that more Web sites are being developed using Ajax practices. Ajax provides the technology that allows a Web site or Web-based application to communicate with the server without having to refresh the entire page. Essentially, the asynchronous features provide the means for the client browser to send requests or call methods that are executed on the server side. The result from the server can then be processed on the client side using JavaScript code, and any output can be merged into the existing front-end HTML view without having to refresh the page. When you use Ajax, you're not really using a new programming language. In fact, all you're doing is taking advantage of existing technologies and putting them to better use.

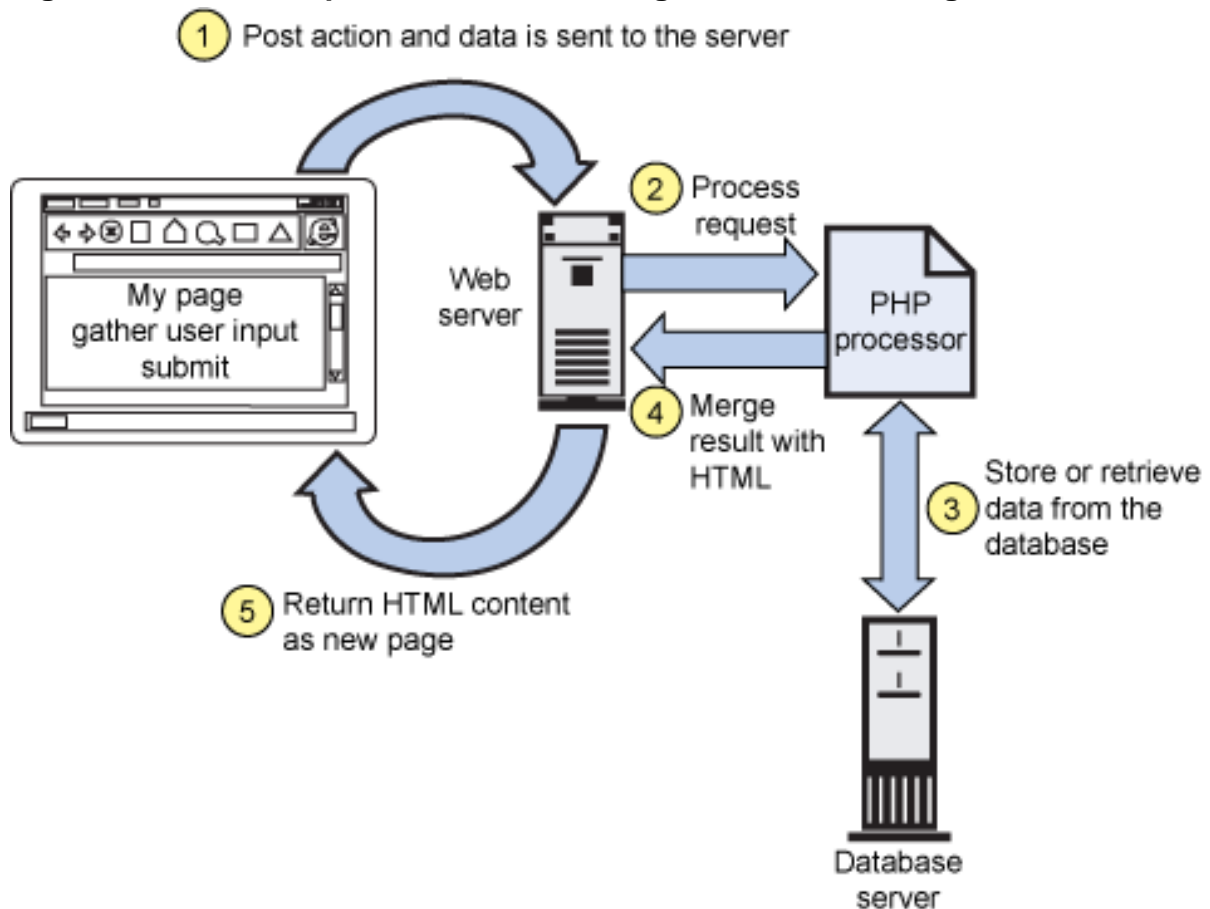
When combined, PHP and Ajax provide a powerful platform for creating Web sites or Web-based applications with robust features. This article looks at some of the uses for PHP and Ajax and examines how you can take advantage of them in your Web-based applications. Before you proceed, a solid understanding of HTML and

JavaScript code is essential. You should also be familiar with PHP as a scripting language, although most any scripting language can be substituted.

Communicating with the server

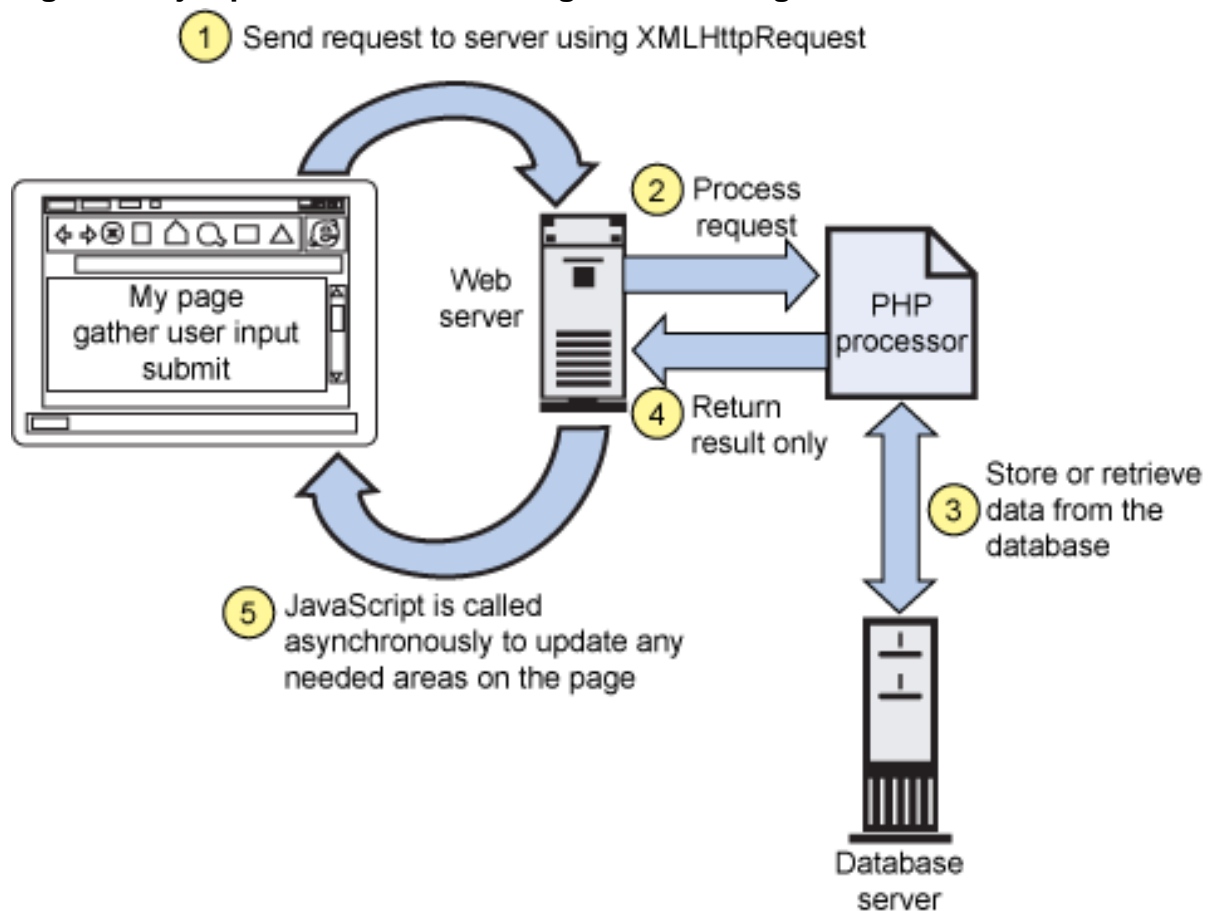
The key to client/server communication in Ajax is to use the JavaScript XMLHttpRequest object. This object is supported by most browsers, including Windows® Internet Explorer 5.0 and higher, Safari 1.2, Mozilla Firefox, Opera 8 and higher, and Netscape 7. To understand the difference between traditional client/server communication and Ajax-based client/server communication, I'll use an example. Traditionally, for the client browser to send content to the server for processing or storing in a database, you usually use a POST action to send content from input fields collected on the client side to the server. The server processes this content using PHP (or any scripting language of your choice), reads or stores data using a database, and returns the results embedded within HTML code. The HTML is then processed by the browser and a new page is rendered for the end user to view. Figure 1 depicts this scenario.

Figure 1. Traditional process for submitting data and receiving results



Using Ajax, the same process requires less time on the front end. The idea is to make users feel as though they never have to wait for a page to update. In fact, using Ajax, you can develop the entire Web application using a single HTML page, though I highly recommend you don't. Traditionally, if you want to send a form to the server, you set the action of the form and specify the action type as `POST`. With Ajax, you don't actually submit a form directly to the server. Instead, you call a JavaScript function that verifies and collects the values from your form and then sends the data to a server-side function using `XMLHttpRequest`. The result is sent back at some point to the client, which then processes the result and updates the portions of the page that need updating. In this case, the page is *not* entirely refreshed. Therefore, less time is spent processing HTML. As a result, the performance is better. Figure 2 illustrates how the process is slightly different when using Ajax, producing an update to the page rather than a refresh of the entire page.

Figure 2. Ajax process for submitting and receiving results



Look at the steps needed to make communication with the server a reality from your JavaScript code. First, you define what the form will look like, as shown in Listing 1.

Listing 1. The HTML form

```
<body>
My First Ajax Page

<form name="myForm">
Press button to view server time:
  <input type="button" value="Update"
    onClick="ajaxFunction();" />
Server Time Is: <input type="text" name="time" />
</form>

</body>
```

This form produces the output shown in Figure 3.

Figure 3. The HTML form's output

My First Ajax Page

Press button to view server time: Server Time Is:

The form doesn't do anything really useful, but hopefully it helps you start seeing the places where you could integrate Ajax into your own code.

Notice the `onClick` event specified on the button. This event is set to call a JavaScript function named `ajaxFunction`. Here's where things get interesting in the Ajax world. Within this method, you perform a few steps, which are explained in this section:

1. Create an instance of the required `XMLHttpRequest` object.
2. Open a connection to the server-side service you want to call.
3. Tell Ajax which method to call when the server-side code completes and returns the result.
4. Send the request.
5. Respond asynchronously.

Creating an `XMLHttpRequest` instance

You need to create the `ajaxFunction` and provide a variable to hold the `XMLHttpRequest` object when it's created. As with any JavaScript method, you define the method as shown in Listing 2.

Listing 2. Definition of the `ajaxFunction`

```
function ajaxFunction() {
    var xmlhttp = null;
    .
    .
}
```

Most modern browsers support the `XMLHttpRequest` object natively. However, older browsers such as Internet Explorer 6 require that you create an ActiveX object to perform asynchronous calls to the server. This poses a problem because you have to determine which browser your code is running in and create the correct object for the job. JavaScript code provides a solution with its support for `try/catch` functionality. You simply try to create the objects in order of preference and let the `try/catch` block handle the rest, similar to the code provided in Listing 3.

Listing 3. Creation of the appropriate `XMLHttpRequest` object

```
function ajaxFunction() {
    var xmlhttp=null;

    try
    {
        // Firefox, Internet Explorer 7. Opera 8.0+, Safari.
        xmlhttp = new XMLHttpRequest();
    }
    catch (e)
    {
        // Internet Explorer 6.
        try
        {
            xmlhttp = new ActiveXObject("Msxml2.XMLHTTP");
        }
        catch (e)
        {
            try
            {
                xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
            }
            catch (e)
            {
                return false;
            }
        }
    }
}
```

As you can see, all modern browsers are written to support the native creation of the `XMLHttpRequest` object. There is some debate as to whether Microsoft® is truly supporting the `XMLHttpRequest` object natively or simply wrapping the ActiveX implementation using a façade.

Opening a connection to the server

Within the same JavaScript function, following the creation of the `XMLHttpRequest` object, you proceed by opening a connection to the server-side script using the `open` method of the `XMLHttpRequest` object. This method receives two required parameters and three optional parameters, which are described in Table 1.

Table 1. Parameters of the open method

Parameter	Description
method	Specifies the HTTP method that you want to use. Possible values include <code>GET</code> , <code>POST</code> , <code>PUT</code> , or <code>HEAD</code> .
url	Specifies the absolute or relative path to the XML data or server-side XML Web service you want to call. To prevent cross-site scripting attacks, Ajax requests can only be made to URLs with the same protocol, host, and port as the page containing the Ajax request. Although some browsers might allow arbitrary URLs, you shouldn't rely on this support from all browsers. If you require cross-site communication, it must be handled on the server side using <code>cURL</code> or some other means.
async	True if you want to send the request to the server asynchronously. A value of true also requires that you set the <code>onreadystatechange</code> property, which I'll discuss shortly. Setting this value to false prevents most browsers from receiving any further input from the user. If your application is flexible enough to continue to receive input while the back-end operation completes, it is best to perform your action asynchronously.
user	Specifies a user name to be used for the purpose of authenticating the user before the script is executed. This is needed only if the script requires user authentication.
password	Specifies a password to be used for the purpose of authenticating the user before the script is executed. This is needed only if the script requires user authentication.

For this example, the code only needs to perform a `GET` operation, requesting the time from a script on the server. You also tell the `open` method that you want to perform the operation asynchronously, as shown in Listing 4.

Listing 4. Call to the open method

```
function ajaxFunction() {  
    :  
    :
```

```
.  
xmlHttp.open("GET", "time.php", true);  
}
```

Telling Ajax which method to call when the server-side code completes

When you call the server using Ajax, the resulting response calls back through a callback function. You can create and name a function or create a nameless function, as I've done. In either case, you need to tell the `XMLHttpRequest` object which callback function to use by setting its `onReadyStateChange` property, as shown in Listing 5.

Listing 5. Setting the `onReadyStateChange` property

```
function ajaxFunction() {  
.  
.  
xmlHttp.onreadystatechange=function() {  
    if(xmlHttp.readyState==4)  
    {  
        // Get the data from the server's response.  
        document.myForm.time.value=xmlHttp.responseText;  
        xmlHttp=null;  
    }  
}  
}
```

As you can see, the method checks the `readyState` after it is called, looking for a possible value of 4. There are five possible states, which are described in Table 2.

Table 2. Possible values for `readyState`

Value	Description
0	Uninitialized
1	Loading
2	Loaded
3	Interactive
4	Complete

The code is essentially saying, "If the state says that the operation is complete, then proceed." When the state is complete, the next step is to update the portion of the page that needs updating with the response from the server. This is done by retrieving the value assigned to the `responseText` property, which is filled with the response from the server. Finally, you stop the `XMLHttpRequest` object by assigning it to `null`.

Sending the request

There is only one more step that the `ajaxFunction` has to perform, and that is to send the request to the server. This is done with the `send` method of the `XMLHttpRequest` object. If the request is asynchronous, this method returns immediately after sending the request. If it is synchronous, the method returns only after the response is received, which means that the Ajax function blocks until the method returns.

This method has one parameter that you can set to null or a number of other values. For example, you can pass a `DOMDocument` object, an `InputStream`, or a `String`. The value is used as the body of the HTTP request if the request method is `POST`. The complete `ajaxFunction` should resemble Listing 6.

Listing 6. The complete `ajaxFunction`

```
function ajaxFunction() {
    var xmlhttp=null;

    try {
        // Firefox, Internet Explorer 7, Opera 8.0+, Safari
        xmlhttp = new XMLHttpRequest();
    } catch (e) {
        // Internet Explorer 6.
        try {
            xmlhttp = new ActiveXObject("Msxml2.XMLHTTP");
        } catch (e) {
            try {
                xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
            } catch (e) {
                alert("Your browser does not support AJAX!");
                return false;
            }
        }
    }
    xmlhttp.open("GET", "time.php", true);
    xmlhttp.onreadystatechange=function() {
        if(xmlhttp.readyState==4) {
            // Get the data from the server's response.
            document.myForm.time.value=xmlhttp.responseText;
            xmlhttp=null;
        }
    }
    xmlhttp.send("");
}
```

Responding asynchronously

Because Ajax development is asynchronous by nature, you must be careful with concurrency and mind your processing sequence. This is especially important when you make multiple Ajax calls and receive the responses in an unexpected order. The rule of thumb is to expect the unexpected.

There may be times when the response from the server needs to come back to the

client as an XML document. In these cases, you can use the `responseXML` property of the `XMLHttpRequest` object to retrieve the data. Listing 7 demonstrates what the PHP script might look like.

Listing 7. PHP code that returns an XML document

```
<?php
$result = getRecordSet($_GET['query']);

echo '<?xml version="1.0" encoding="ISO-8859-1"?>' .
    '<car>';

while($row = mysql_fetch_array($result)) {

    echo "<make>" . $row['make'] . "</make>";
    echo "<model>" . $row['model'] . "</model>";
    echo "<year>" . $row['year'] . "</year>";
    echo "<description>" . $row['description'] .
        "</description>";
}
echo '</car>';
?>
```

As you can see, the code creates an XML document, denoted by the `<?xml version="1.0" encoding="ISO-8859-1"?>` line, prior to sending the rest of the response. The client-side code must be coded so that it first pulls the XML document from the `responseXML` property, parses the data using the Document Object Model (DOM), and then displays the data on the page by modifying the appropriate fields that are already on the page. Listing 8 demonstrates these actions.

Listing 8. JavaScript code that processes an XML document

```
function stateChanged() {
    if(xmlHttp.readyState==4) {
        xmlDoc = xmlHttp.responseXML;
        document.getElementById("make").innerHTML =
            xmlDoc.getElementsByTagName("make")[0].childNodes[0].nodeValue;
        document.getElementById("model").innerHTML =
            xmlDoc.getElementsByTagName("model")[0].childNodes[0].nodeValue;
        document.getElementById("year").innerHTML =
            xmlDoc.getElementsByTagName("year")[0].childNodes[0].nodeValue;
        document.getElementById("description").innerHTML =
            xmlDoc.getElementsByTagName("description")[0].childNodes[0].nodeValue;
    }
}
```

Extending Ajax with JavaScript frameworks

You've probably noticed that there is nothing special needed from the server-side PHP code to implement Ajax functions on the front end. In fact, you can actually

write the server-side script in any language that can be called from your Web server and return its result to the client. However, there are several initiatives to extend Ajax to do other things. For example, there are a number of libraries that provide much more visually appealing front-end controls to make the dream of Web 2.0 a reality. Most of these libraries are written in JavaScript code and provide features and APIs that decrease the amount of JavaScript code you need to write on the front end. Also common to most of these libraries is the use of JavaScript Object Notation (JSON), which is a lightweight data-interchange format used all over the Internet today. The following are some of the libraries that you might find helpful:

- Prototype
- Script.aculo.us
- Dojo
- jQuery

Prototype

Prototype provides class-driven JavaScript code for developing dynamic Web applications. Additionally, the Prototype framework provides a global Ajax object that makes your Ajax functions easy to write and, even better, easy to process. Using this framework for your Ajax development provides the added advantage that you no longer have to deal with the cross-browser issues. With one call, you can provide the destination URL, the HTTP method (`POST` or `GET`), the response callback method, and much more.

Script.aculo.us

Another popular JavaScript framework with some nice user interface controls is Script.aculo.us. This framework provides some controls that perform Ajax functions for you. There's even a Google-like auto-complete text control that you can use to collect text from your users. The Ajax functions allow you to easily specify the server-side target of the query. The resulting response is used to display a set of strings returned from the server to the end user as possible choices.

Dojo

The Dojo toolkit is an open source JavaScript and Ajax framework, which is popular for its speed and 25 KB size. In the Ajax arena, Dojo provides its own APIs for sending and receiving asynchronous calls to and from a server. On the server side, nothing changes. Your PHP code continues to respond as it always has to any Ajax calls.

jQuery

jQuery does a great job of simplifying the mechanism for search for elements in your pages and adding new functions, features, and formatting on the fly at runtime. It also includes some distinctive Ajax features that allow you to provide Ajax interactions on your Web pages.

Extending PHP To support Ajax

Now that you understand the basics, I'll show you how PHP is being extended today to support Ajax. From a PHP perspective, extending Ajax isn't just about rewriting the `XMLHttpRequest` (XHR) access code. It's about providing front-end services that can access back-end services written in PHP in a much simpler and more intuitive manner. Services such as executing back-end services, accessing a database, and performing back-end network services from the front end should be included. Some of the frameworks I've seen on the market allow developers to write their code in the back-end language, and the framework creates the front-end JavaScript code. Essentially, you tell it what back-end methods must be called, and the framework writes front-end façades in JavaScript code that can allocate an XHR object, send the request, receive the response, and then pass it to a function of your choice or even assign the output directly to an HTML element of your choice. The idea is to decrease the amount of JavaScript code that you have to write. With this in mind, all elements that you want to provide functions for can be overridden and tied to back-end methods that react when a specified front-end event occurs. The following are frameworks worth investigating:

- PHP AJAX
- PHP-Ext
- ExtPHP

PHP AJAX

PHP AJAX allows you to extend a PHP class with your own class. Then you simply call an initialization method when your PHP file is called, and it generates the necessary front-end XHR code for you. The front-end event must then call a JavaScript function with the same name as your PHP class name. The code looks similar to Listing 9.

Listing 9. Extending PHP AJAX with your own PHP class

```
class ajax_app extends phpajax {
    function input() {
    }
    function loading() {
    }
    function main() {
    }
}
```

```
}
```

The overridden methods are called to perform various tasks in the Ajax life cycle. For example, if an action is executed on the front end, the back-end class's `main()` method is called to handle the action. Any input collected on the front end is passed to the `input()` method.

PHP-Ext

PHP-Ext provides a library written to support both PHP 4 and 5 that provides an abundance of front-end user interface controls. The underlying base of these controls is actually provided by Ext JS, but what makes this framework unique is that you don't have to concentrate on providing JavaScript to manipulate the front-end controls. Instead, all of the interaction with the controls is performed using back-end services written in PHP. The PHP code is called as needed, depending on which events you specify you want to be called for. When a registered event occurs, the matching PHP method is called in your code. This way, you can fill the content of grids and other controls almost directly from your PHP code.

ExtPHP

Another wrapper for Ext JS written in PHP is ExtPHP (not to be confused with the previously mentioned PHP-Ext). According to the project's lead developer, you can use PHP-Ext write intrusive and non-intrusive JavaScript code, just as you would with Ext JS. The advantage is that this design allows your PHP editor to detect unknown, misspelled, or misused methods rather than forcing you to debug the JavaScript code in the Web browser. The idea behind ExtPHP is to generate PHP classes that can be called from your PHP scripts. When you instantiate an object, the corresponding JavaScript code is stored in an internal buffer. If you call an object's method, the code is added to the internal buffer. When you're ready to complete the JavaScript code, you simply call a single method, `jsrender()`. You can also add straight JavaScript code to your output by calling ExtPHP functions that allow you to add to the output.

Summary

This article provided you with an introduction to writing Ajax-based front-end code and then showed you how to combine the front-end code with back-end PHP scripts. It also directed you to some of the latest available frameworks that allow you to create Ajax-based Web applications much faster than if you write all of the code yourself. By combining one or two of these frameworks with Ajax and PHP, you can write rich and visually appealing Web applications. In fact, you may wonder how you ever wrote your Web applications without these newer technologies.

Resources

Learn

- [JavaScript.com](#) is a great resource for learning about JavaScript code and upcoming technologies related to it.
- The [PHP Ajax Frameworks](#) Web site provides a list of frameworks for developing PHP/Ajax applications without having to perform all of the Ajax JavaScript coding yourself. Many of these frameworks perform the work for you in the background, allowing you to concentrate on the guts of your application, rather than the infrastructure to achieve Ajax client/server communications.
- "[Ajax & PHP without using the XMLHttpRequest Object](#)" has some really cool ideas, describing how you can write PHP code that allows you to use Ajax concepts without having to depend on the `XMLHttpRequest` object. However, it is limited to the `GET` method, so if you need to send the server more than 500 bytes, you might have to find another means to do so.
- For all things Ajax, [Ajaxian](#) is definitely the place to go. This is a must-see Web site. Everything that has anything to do with Ajax is on this Web site. Whether it has to do with PHP or some other language, you'll find it here.
- Expand your Web development skills with articles and tutorials that specialize in Web technologies in the developerWorks [Web development zone](#).
- Stay current with developerWorks' [Technical events and webcasts](#).

Get products and technologies

- If JavaScript was loaded with an upgraded API or Framework, it would be [Prototype](#). Where the JavaScript API ends, this one picks up. This framework has already become the standard used by other JavaScript framework developers as the underlying basic framework on which to build new ones. In fact, [script.aculo.us](#) was built on this one.
- The [script.aculo.us](#) library provides some of the nicest UI controls I've seen. I've walked away from many happy clients thanks to this handy little library.
- [JSON](#) is a lightweight data-interchange format that is readable by humans and easily parseable by computers. Not just for JavaScript code anymore, JSON has been added to the [PHP language in version 5.2.0](#).
- [Ext JS](#) is a JavaScript framework with Ajax and UI components. It works in conjunction with Prototype, Yahoo UI, and jQuery. There are both commercial and open source licenses available for this project.
- [Dojo](#) provides Ajax, events, packaging, CSS-based querying, and a lot more in a neat little package.

- [jQuery](#) is a JavaScript library that simplifies how you traverse HTML documents, handle events, and perform animations. It also has some Ajax functionality that allows you to easily add Ajax to your front-end HTML for calling back-end server-side scripts.
- [PHP AJAX](#) is a framework that generates all of the needed JavaScript code on the fly after the PHP file is called. You write a PHP class that extends a PHP Ajax class and overrides the methods to handle the front-end calls on the back end.
- [PHP-Ext](#) is an open source widget library based on Ext JS that provides developers the ability to write PHP scripts instead of JavaScript code.
- [ExtPHP](#) is a library you can use to write intrusive and non-intrusive JavaScript, just as you would do with Ext JS.

Discuss

- Check out [developerWorks blogs](#) and get involved in the [developerWorks community](#).

About the author

Ken Ramirez

Ken Ramirez is the founder of [Axsys Technology Group](#), which provides hosting and template or custom Web site development through [AxsysHosting.com](#). Ken also works with small business owners looking for inexpensive Web site solutions through his [BuildMySiteTonight.com](#) Web site. His company specializes in PHP, MySQL, Linux, Windows, XHTML/CSS, Flash, E-Commerce solutions (such as osCommerce and osCMax), and Content Management Solutions (such as Joomla). You can reach Ken via <http://axsyshosting.blogspot.com>.

Trademarks

Java and all Java-based trademarks and logos are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.