AJAX Example

For this discussion we need to refer to FIGURES 8.5 and 8.6 for the markup and code, and to FIGURE 8.7 for the corresponding browser display. We begin with a high-level view. In Figure 8.5 we have a script element within the head element. This script element contains the definitions of the two JavaScript functions getCurrentTime( ) (lines 25–32) and updatePage( ) (lines 33–42) that we call later on in the second script that appears at the end of the body element in our document (lines 61–64).

Now let’s itemize the things that take place when this page loads:

As in the previous example, we start a PHP session and initialize a counter (lines 12–14). Note that this time around we call our counter key timedateRefreshCount rather than the former pageRefreshCount because in this example it is only the time and date information that will be refreshed, not the entire page.

The script element in the head element of the document defines two JavaScript functions that we call later.

The complete page is displayed for the first (and only) time by lines 46–60. This display includes the content produced by the short PHP script in lines 47–50, which we have discussed previously. Note in passing, however, that the value of the id attribute of the h3 element in the PHP script output is "timedate", since we will want to refer to this later.

Now we come to the script element in lines 61–64. The first thing this script does is call the JavaScript getCurrentTime( ) function, defined above in lines 25–32. This function, in turn, begins by creating an object reference of type HttpXmlRequest (line 27) and assigning it to the variable request. This is the object that will make our AJAX request to the server. To do this it needs to know the name of the script on the server to which it should make its request. We supply the name of the script to the variable url in line 28 and use this variable as one of the parameters in the function call on the next line. There is a bit of subtlety in calling the variable url, even though its value is just the name of a PHP script file. We can use only the file name because that script file is located in the same directory as the web page from which it is being called. But it could be a script that is located elsewhere at a location that would have to be given as a “real” URL for the value of the url variable.

Next (line 29) a call to the request.open( ) function opens a connection to the script at the given url value (the second parameter), indicating the method for passing the data will be the GET method (the first parameter in the function call, and discussed in detail in section 8.6). The third parameter in the function call is the boolean value true, which in this case indicates that the connection is to be asynchronous. This means that once the connection has been made, the browser can carry on with other business and doesn’t have to wait until the connection is finished doing whatever it has to do and closes.

The browser needs to know what to do when it hears back from the server. In this case we assign to the request.onreadystatechange property of our request object a reference to our JavaScript updatePage( ) function (line 30). But note that there are no parentheses following the function name in line 30 because we are not calling the function at that point. The function will be called only when the “ready state” of the function changes, in other words only when the browser hears back from the server.

Finally, now that we are all set up, the AJAX request is sent to the server in line 31. The parameter is null in request.send(null) because we have no additional information to send to our server-side script.

Let’s shift to a server-side view and look at the server-side PHP script time.php shown in Figure 8.6. There is really nothing here we haven’t seen before; we have just grouped, consolidated, and rearranged some of the PHP code shown in Figure 8.3. But let’s describe what it does in this new context.

The script begins by “joining” the PHP session that has already been started by lines 12–14 in Figure 8.5. Since this is the first call to the time.php, the “time date refresh count” will be 0, so the “greeting color” will be reaffirmed to be black. The date and time are then computed and sent back to the browser, along with the hidden paragraph containing the text color. Then, just before the script ends, the count is incremented so that on subsequent runs of this script it will not be 0, and a random color will be chosen for the text of the date and time display.

Now we go back to what’s happening in the browser and look again at Figure 8.5. When the time.php script sends its information back to the browser, the “ready state” of the request object will change, and that’s when our second JavaScript function update Page( ) is called. The body of this function (lines 35–41) consists of a single if-statement that tests the readyState value of the request object. It can have several values, but we want it to have the value 4, which essentially means “everything’s OK”. In that case the body of the if-statement is activated, so we get a reference to the h3 element with id value "datetime" and change its text color to whatever color came back from the server as the content of the hidden paragraph.

The call to the global JavaScript function setInterval( ) in line 63 of Figure 8.5 says, “Call the getCurrentTime( ) function every 60000 milliseconds (every minute), from now on.” Henceforth, then, the whole process will repeat, and from now on some color other than black will be used. Note, however, that although a “new” random color is chosen every minute, because there are only four colors from which to choose it may well be the case that the same color appears for two (or even more) intervals in a row.

More information:

PHP Date Function [http://www.w3schools.com/php/func\_date\_date.asp](http://www.w3schools.com/php/func_date_date.asp%20)

W3 AJAX Tutorial [http://www.w3schools.com/ajax/default.asp](http://www.w3schools.com/ajax/default.asp%20)

AJAX PHP DB example <http://www.w3schools.com/php/php_ajax_database.asp>