Lab 7 – Introduction to Ajax

Create a new folder '**lab07**' on the cmslamp14 server
Save today's work in this 'lab07' folder.

Task: A simple Ajax application (10 marks)

❑ The overall task is to get a general idea of how Ajax works and modify program to incorporate additional concepts

Step 1:
Copy from Blackboard, into your 'lab07' folder on the server the various sample files from Lecture 7 which are stored in the "Week 7" folder, within the "Lectures" section on AUTonline. Load them into your preferred editor, and read the files carefully, so that you get a good feel for the structure of the example, and appreciate how the html, Javascript and PHP technologies work together. In both IE and Chrome, go to the URL of the HTML file (simpeajax.htm), and run the example. The php program has a "sleep" statement in it, so that the server takes time to respond. Experiment with various client interactions in the waiting period – for example, try entering different data whilst waiting for the server to respond, and see what the eventual response is.

Step 2:

Modify the example so that it uses the POST rather than the GET protocol. (This is more suitable for a user id/password submission, for security reasons.)

The code for this is given in the Appendix. We will explain this in later in the course.

Step 3:

THIS EXERCISE WILL BE OF MAJOR HELP TOWARDS THE FIRST ASSIGNMENT. IT IS LONG, BUT THE WORK DONE WILL PAY OFF.

- Modify the example further so that it does something a little more meaningful. Set up a simple MySQL database with one table, that stores names, passwords and email addresses. The "name" should be the primary key, so there are no duplicate entries
- Populate the tables (using MySQL commands) with some test data (see below).
(Note: In a real application, the data in these tables will be managed by a separate web program that allows an administrator to maintain the data. In our example we are only exploring the user interaction to search for an email address, and so we will work with a table whose data has been input directly using MySQL commands.)
- Change the PHP files so that it (i) checks that the password is ok for the name, and (ii) if so, it returns the email address to the client, where it is displayed nicely.
- You should deal properly with the possible "error" conditions: (a) the name is not in the table and (b) the name is in the table, but the password is wrong. Suitable text messages should be sent back to the client, and displayed, in each of these cases. {You may choose to use the error message sent back to cause the client to display an alert (using JavaScript), rather than writing to the output field in the form.}
-
- Run the application, and use suitable test cases. Document the test cases that you use. {Make sure that you test all of the possible classes of outcomes – ie name in database and password correct, name in database and password wrong, and name not in database.}

APPENDIX : USING POST

```
// file simpleajax.js
// using POST method
var xhr = createRequest();
function getData(dataSource, divID, aName, aPwd)  {
   if(xhr) {
           var obj = document.getElementById(divID);
           var requestbody ="name="+encodeURIComponent(aName)+"
                     &pwd="+encodeURIComponent(aPwd);
           xhr.open("POST", dataSource, true);
           xhr.setRequestHeader("Content-Type", "application/x-www-form-
urlencoded");
           xhr.onreadystatechange = function() {
             alert(xhr.readyState); // to let us see the state of the computation
             if (xhr.readyState == 4 && xhr.status == 200) {
                obj.innerHTML = xhr.responseText;
              } // end if
           } // end anonymous call-back function
           xhr.send(requestbody);
        } // end if
} // end function getData()
```

- Note that the **request body**, contains the name/value pairs.
- We need to **URI encode** the data being passed.
- We need to **open xhr for POST**ing (rather than GETting).
- We need to **set the Request Header**.
- We need to **send the request body to xhr**.
- As with the example in lectures, we include a statement to show the state of the readyState property of xhr each time that state changes. This would not be done in a real system (for example, when you submit an assignment), but it is very useful to help us understand how Ajax works.