



# METAOPTION

An Article  
on  
Ajax

## Using AJAX in real life scenario

AJAX is an acronym for **Asynchronous JavaScript and XML**. If you think it doesn't say much, we agree. Simply put, AJAX can be read "empowered JavaScript", because it essentially offers a technique for client-side JavaScript to make background server calls and retrieve additional data as needed, updating certain portions of the page without causing full page reloads. Figure 1.4 offers a visual representation of what happens when a typical AJAX-enabled web page is requested by a visitor:

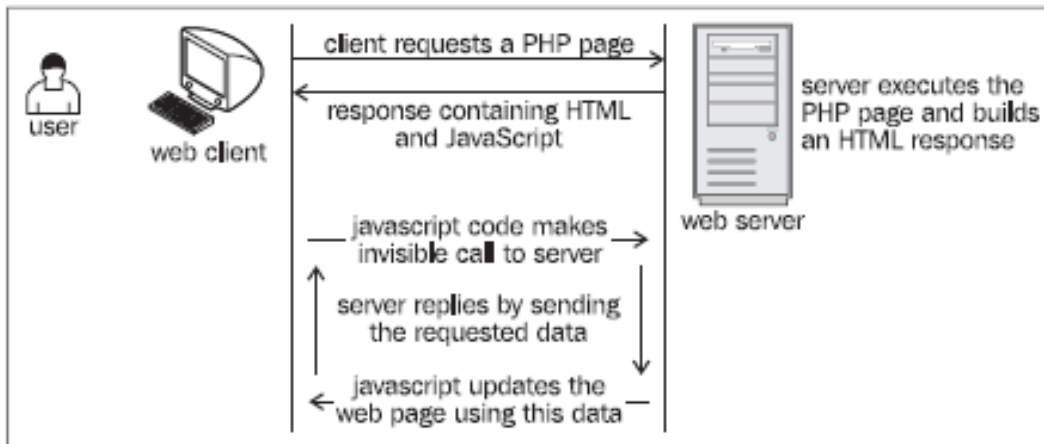


Figure 1.4: A Typical AJAX Call

When put in perspective, AJAX is about reaching a better balance between client functionality and server functionality when executing the action requested by the user. Up until now, client-side functionality and server-side functionality were regarded as separate bits of functionality that work one at a time to respond to user's actions. AJAX comes with the solution to balance the load between the client and the server by allowing them to communicate in the background *while the user is working* on the page.

**I have used AJAX in different scenarios; I am listing some of the problems in which how I have used AJAX:**

**Problem 1:** We had a project which is related to tracking the loans online. In this project on one page client ask to sort the loan documents, but he doesn't want to do this in old fashion, he wants user to drag and drop the documents at desired position and then automatically documents will be sorted on the basis of their current position.

**Solution:** I was really confused how to achieve drag and drop with database updating.

I know the JavaScript and can make the page components draggable but the main problem was to achieve the database updating at the same time. But thankfully we have a Jeanie called AJAX .AJAX provides the facility to interact with server without the post back. And really want to

thanks the AJAX library developer who has written various libraries of AJAX (basically set of JavaScript functions). In market there are lots of free AJAX libraries are present which can make the development fast and efficient for e.g. YUI (yahoo user interface) library, Google library, scriptaculous library etc.

So we have chosen to accomplish our goal with the help of scriptaculous library.

So we have created a table and populate its rows with data. On client side we have already define the rows as draggable. And we saved the three javascript files (prototype.js, scriptaculous.js and dragdrop.js) of scriptaculous library in our javascript directory,

We have not taken all the files of the scriptaculous library because we don't want to waste the server space and we have taken only three files that we have needed.

After saving, now our work is to make the table sortable by using Sortable.Create function of scriptaculous library. Now our table is ready with drop and down facility.

On every drag and drop most of the work is handle by Serialize() and Process() function of scriptaculous library. Actually Serialize() function serialize the id's of the table content and make an array and Process() function pass this updated sorted list to an aspx page through ajax where actual database updation take place.

I am pasting some of the screenshots of our sortable table with draggable rows.









|   |   |
|---|---|
|  CONSTRUCTION/PROJECT INFORMATION                                | <a href="#">[Clone Section]</a>   <a href="#">Add Document</a>   <a href="#">Delete Section</a>   <a href="#">Details</a> |
|  FINANCING SOURCES  | <a href="#">[Clone Section]</a>   <a href="#">Add Document</a>   <a href="#">Delete Section</a>   <a href="#">Details</a> |
|  OPERATING INFORMATION   | <a href="#">[Clone Section]</a>   <a href="#">Add Document</a>   <a href="#">Delete Section</a>   <a href="#">Details</a> |
|  PROFORMAS   | <a href="#">[Clone Section]</a>   <a href="#">Add Document</a>   <a href="#">Delete Section</a>   <a href="#">Details</a> |
|  RENTAL ASSISTANCE   | <a href="#">[Clone Section]</a>   <a href="#">Add Document</a>   <a href="#">Delete Section</a>   <a href="#">Details</a> |
|  ITEMS REQUIRED FOR HISTORICAL PROJECT                         | <a href="#">[Clone Section]</a>   <a href="#">Add Document</a>   <a href="#">Delete Section</a>   <a href="#">Details</a> |
|  PARTNERSHIP DOCUMENTS (Due at closing and for intial funding) | <a href="#">[Clone Section]</a>   <a href="#">Add Document</a>   <a href="#">Delete Section</a>   <a href="#">Details</a> |
|  PERMANENT LOAN DOCUMENTS                                      | <a href="#">[Clone Section]</a>   <a href="#">Add Document</a>   <a href="#">Delete Section</a>   <a href="#">Details</a> |

Fig 1.1: This is our initial table of loan deals.

|   |   |
|---|---|
| + CONSTRUCTION/PROJECT INFORMATION                              | [Clone Section   Add Document   Delete Section   Details] |
| + FINANCING SOURCES   | [Clone Section   Add Document   Delete Section   Details] |
| + PROFORMAS   | [Clone Section   Add Document   Delete Section   Details] |
| + RENTAL ASSISTANCE   | [Clone Section   Add Document   Delete Section   Details] |
| + OPERATING INFORMATION   | [Clone Section   Add Document   Delete Section   Details] |
| + ITEMS REQUIRED FOR HISTORICAL PROJECT                         | [Clone Section   Add Document   Delete Section   Details] |
| + PARTNERSHIP DOCUMENTS (Due at closing and for intial funding) | [Clone Section   Add Document   Delete Section   Details] |
| + PERMANENT LOAN DOCUMENTS                                      | [Clone Section   Add Document   Delete Section   Details] |

Fig 1.2: Now we are dragging operating information from 3<sup>rd</sup> to 5<sup>th</sup> position.

|   |   |
|---|---|
| + CONSTRUCTION/PROJECT INFORMATION                              | [Clone Section   Add Document   Delete Section   Details] |
| + FINANCING SOURCES   | [Clone Section   Add Document   Delete Section   Details] |
| + PROFORMAS   | [Clone Section   Add Document   Delete Section   Details] |
| + RENTAL ASSISTANCE   | [Clone Section   Add Document   Delete Section   Details] |
| + OPERATING INFORMATION   | [Clone Section   Add Document   Delete Section   Details] |
| + ITEMS REQUIRED FOR HISTORICAL PROJECT                         | [Clone Section   Add Document   Delete Section   Details] |
| + PARTNERSHIP DOCUMENTS (Due at closing and for intial funding) | [Clone Section   Add Document   Delete Section   Details] |
| + PERMANENT LOAN DOCUMENTS                                      | [Clone Section   Add Document   Delete Section   Details] |

Fig 1.3: Now we have updated table with “operating information” deal at 5<sup>th</sup> position.

**Problem 2:** It is the general problem that we were facing from early age of internet, if you can remember yahoo sign up page, it shows the “username already exist” message when user fill up the whole form data and the page post back and then after a minute wait it shows “username already exist”. In our new project which is related to business listing,

In admin part we had a goal to check all the errors related to duplicity in client side only means without the post back.

**Solution:** The only solution that I have got to achieve this goal is AJAX only. We want to check the database for duplicity without posting the page back to the server. AJAX provides this facility to interact with database through JavaScript. So we started this with some JavaScript functions. In first function we have initiated the XMLHttpRequest object.

```
function Initialize()
{
    try
    {
        req=new ActiveXObject("Msxml2.XMLHTTP");
    }
    catch(e)
    {
        try
        {
            req=new ActiveXObject("Microsoft.XMLHTTP");
        }
        catch(oc)
        {
            req=null;
        }
    }

    if(!req&&typeof XMLHttpRequest!="undefined")
    {
        req= new
        XMLHttpRequest();
    }
}
```

Now we need a function that will interact with code behind page. So we have designed a function called SendQuery in which we pass the URL of the page on which function of database duplicity has been written, along with the querystring parameters. In this function first we initialize the XMLHttpRequest object then we call the function process which will send the query to the page and collect the resultant data as xml or simple text, I will explain the process function in next step.

```
function SendQuery(URL, divID)
{
    //append "JS" as querystring in URL
    URL = URL + "&Caller=JS";
    gDivID = divID;
    Initialize();
    if(req!=null)
    {
        req.onreadystatechange = Process;
        req.open("GET", URL, true);
        req.send(null);
    }
}
```

Now we have process function in which we wait for output xml or text file until we get "OK" message from browser (status 200 represent Ok)

As soon as we get the output xml/text we all take it and populate the desired div with the generated output, in our case we are paasing the id's of the div in which I am showing "already exist" message.

```
function Process()
{
    if (req.readyState == 4)
        {
            // only if "OK"

            if (req.status == 200)
                {
                    var responseString = '';
                    responseString = req.responseText;
                    if(responseString != '')
                    {
                        document.getElementById(gDivID).style.display = "";
                        if (document.layers[document.layers[gDivID]].innerHTML == responseString;
                            else
                                document.getElementById(gDivID).innerHTML = responseString;
                    }
                    else
                    {
                        document.getElementById(gDivID).style.display = "none";
                    }
                }
        }
}
```

Now our javascript has finished, only the part left is to call the sendquery() function on page and pass the appropriate URL and div id on the event like onblur().

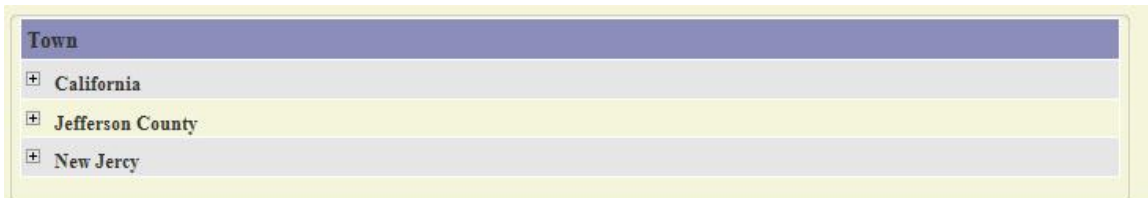


Fig 2.1: Here we have checked the street name duplicity through ajax.

**Problem 3:** Now I want to use one of the great thing that AJAX provide, means populating the only main records on first load of page and populating sub records on request so as to improve the performance of the page. In one of our project in admin

section we have to display all the streets in the database and we have thousands records of street in database and displaying all the streets in gridview definitely results in performance loss.

**Solution:** So in order to solve this problem, we planned to show only town names in gridview, we have 100s of towns in database, so there is no much performance loss. We have planned to fill the street data table through AJAX, on click of town name. We have hid a div along with every town in grid view and on click on town name function we are calling the same send query function that I have shown in problem 2, and we are passing the desired page URL (which returns the table of streets) along with town id as query variable, so at desired URL we can get the streets for particular town on the basis of town id and after getting the value we fill the div along with clicked town with the street table.



| Town               |
|--------------------|
| + California       |
| + Jefferson County |
| + New Jersey       |

Fig 3: GridView with the town names.



| Town               | Street Name            | Description                                  | Image          | Edit                 | Delete  |
|--------------------|------------------------|--|----------------|----------------------|---|
| - California       | street california      | this is a very famous street in california   |                | <a href="#">Edit</a> |  |
|                    | test california street | test california streettest california street |                | <a href="#">Edit</a> |  |
|                    | test california        | test california                              | imagesCAIZD8SM | <a href="#">Edit</a> |  |
| + Jefferson County |                        |  |                |                      |   |
| + New Jersey       |                        |  |                |                      |   |

Fig 3: After clicking "California" the street of california appears.

So now we have seen how can we achieve great performance through ajax just by populating the page with desired result at the request time. For e.g. what will be the use of displaying all the street of all the towns if the user is interested in california town's streets. It also makes the page shorter with ability to search anything(e.g. any text) on page easily.

**Problem 4:** This problem is related to the above problem, in the above feature we wanted to show the image when user mouse over the image name, in street table in database we have only image id and image URL is in another table called images.

**Solution:** To achieve this goal we have two solutions.

- 1) Getting the image URL by joining street table with images at the time of fetching street data for the town(problem 3), and then saving an image control along with every image name in image column and hide it, visible it on mouse over on image name and hide on mouse out.
- 2) Doesn't join any table, and save the image id along with image name and pass the image to particular page (where we get the image from database) through AJAX and populate the div at image name.

We found performance issue in solution number one because joining of table make the query slow and saving images of kilo bytes size on page(in hidden form) making the page bulky and definitely this make the page slower.

So we start working on other solution because there is no joining of table and no saving of image on page. So we have designed a page on which we have defined a function which get the image information from database and set the image URL of the image control on this page with the value from database and then render this control to HTML and write this string on console. Now we come to our streets page here we place a div along with image name and on mouse over to image name we pass this div id and image id along with the URL of desired page (on which we have populated the image control with database value of image URL) to "Sendquery()" and we populate this div with the image control rendered html string that we have generated on desired URL.




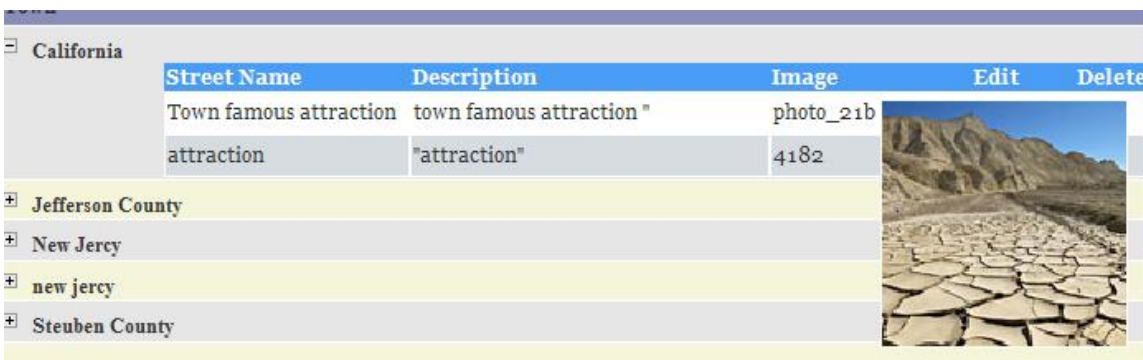
| Street Name            | Description              | Image     | Edit                 | Delete  |
|------------------------|--------------------------|-----------|----------------------|---|
| Town famous attraction | town famous attraction " | photo_21b | <a href="#">Edit</a> |  |
| attraction             | "attraction"             | 4182      | <a href="#">Edit</a> |  |

Fig 4.1: Town with streets information.







| Street Name            | Description              | Image     | Edit  | Delete  |
|------------------------|--------------------------|-----------|---|---|
| Town famous attraction | town famous attraction " | photo_21b |  |  |
| attraction             | "attraction"             | 4182      |  |  |

Fig 4.2: On Mouse over to Image Name we get the original image.

On various other problems we have used AJAX and we are also accessing web services with the help of AJAX in our ongoing project, hopefully in my next article you will find something more complicated, exciting and beautiful.

**SAIF AADIL KHAN**

s/w Engineer, METAOPTION