The International Journal of Multi-Disciplinary Research
ISSN: 3471-7102

Using Array-Shift Algorithm to Implement an AJAX based Statistical Software for Use in Higher Education in Developing Countries

(Conference ID: CFP/168/2017)

Mwenge Mulenga
School of Engineering
Science and Technology,
Mulungushi University,
Kabwe, Zambia,
mwenge2008@yahoo.co.uk

Luckson Simukonda
Application Development,
Rhema Forms Solutions,
Kabwe, Zambia,
thezo1992@gmail.com

Douglas Kunda(PhD)
School of Engineering
Science and Technology,
Mulungushi University,
Kabwe, Zambia,
dkunda@mu.ac.zm

Abstract*–JavaScript based libraries such as JQuery offer a rich user interface on the client side due their ability to interact with the Document Object Model - DOM. When used together with AJAX they come in handy as solution to the stateless nature of traditional web based applications, since objects can be created and manipulated with only partial page updates to and from the server. This advantage can also be a pit fall in that it can allow objects on the client to grow to sizes that are strenuous to both bandwidth as well as server-side processing. In this paper we demonstrate an approach that can be used to optimize usage of these resources especially in resource constrained parts of the world such as Africa. The paper demonstrates that attaining high usability as well as high performance tend to be conflicting requirements, hence the need for a tradeoff. We had to trade of some level of usability so as to attain high performance in terms of bandwidth and server-side computation.*

Keywords*–Array-Rotation, Array-Shift, JQuery, Bandwidth Optimization, Server-Side Computation, Asynchronous JavaScript and XML, AJAX, JSON, Rhema Forms Solution*

I. INTRODUCTION

Research is vital in tertiary level education, and statistical software is a very important part of it. There is a wide adoption of Information Communication Technology – ICT in developed countries for educational use, while developing countries are lagging behind [1]. Consequently, adoption of online data collection and processing software has equally been low in developing countries. Inadequate, Information Communication Technology, ICT infrastructure such as limited internet connectivity and the high other hardware costs are a huge hindrance to the adoption of such applications [2]. The other deterrent to ICT adoption in developing countries is low ICT literacy levels [3].

There is therefore, need to develop applications that consume less resources and are tailored to the literacy levels of the intended audience. Simplified applications can enable students to focus on the problem space rather than the solution space leading to improved performance.

The purpose of this study was to propose the right algorithm and a suitable technique of its implementation in developing the backend of an online data collection software. On one hand the appropriate algorithm and its implementation where supposed to sustain a rich user interactive experience. On the other hand, the study sought to identify an appropriate server side processing algorithm that would circumvent limited processing power available in shared hosting servers. The study also sought to identify an appropriate communication pertain between clients and servers so as to augment the adopted algorithm.

The remainder of this paper discusses related work which is then followed by identification of an appropriate algorithms. We then present test results which are immediately followed by a discussion. In the last chapter we conclude and present recommendations for future works.

II. RELATED WORK

There is a lot of publications that identify poor internet connectivity as a challenge in developing countries. Jay Chen et al [4] points out that intermittent connectivity and low bandwidth is a hindrance to the full utilization of internet by teachers and students in Peri-Urban India. As such they propose how to improve performance by offering a practical tool that accelerates web traffic by using both caching and prefetching. However, they also point out an inclination of users towards dynamic web pages which complicates offline caching.

On other hand there has been a lot headway in developing appropriate technology that optimizes the use of internet resources. Asynchronous data communication offered by Asynchronous JavaScript and XML – AJAX, in web based client/server applications has in the recent past taken center stage. According to [5]AJAX technology improves band width usage in web applications performance due to its ability to partially update a web page. The authors also identify the use of jQuery, a JavaScript based framework, as first choice option for implementing front end logic as its ability to easily manipulate the Document Object Model – DOM and its cross browser compatibility. Based on the above mentioned attributes of AJAX and JQuery the authors propose a general design architecture for asynchronous web applications so as to reduce application complexity and workload.

The use of JavaScript Object Notation – JSON as a data transfer format between client and server improves the performance of web applications [6].

The use of JQuery AJAX calls improves performance in web applications and enhances bandwidth optimization in client/ server communications. Bandwidth utilization is also improved by eliminating overheads of page post back through partial updates. JQuery also supports the creation of a rich user interface similar to desktop applications through its support for DOM manipulations [7].

From the perspective of improved performance through algorithms, Chin-Kuang Shene mentions that Array Rotation algorithms are a popular practice in string and array manipulation in CS1 and CS2 classes [8]. The author also points out that Array Rotation is also used in many common libraries such as STL.

III. ALGORITHM COMPARISONS

In a quest to demonstrate how the right implementation of an algorithm can improve the performance of a rich User Interface AJAX based web application, this paper compares two algorithms namely Array Rotation also referred to as Array-Shift, and a traditional algorithm.

Application Design and Implementation

The application was based on client/server architecture with the server based on PHP and is supported by a MySQL Database Management System. The client side used JQuery with JSON as the data exchange format transmitted through AJAX calls.

Client Side

The client side implements a workspace that allows dynamic creation of an unlimited number of questions by means of drag and drop. The questions, include free input and multiple choice, which can be reordered by way of swapping their positions. This kind of html form elements manipulation utilizes jQuery’s ability to manipulate the DOM allowing for Create, Update and Delete operations commonly referred to as CRUD as well as the above mentioned sort operations. Support of such operations without page post backs create a rich user experience. However, to persist changes made on the client side to the server, there is need for corresponding server side code to be invoked which in turn manipulates that database accordingly. We implemented two ways in which you could save changes. The first method was through clicking the save button for an individual question. The other method was where all questions in questionnaire would be saved by clicking one button("Questions", www.rhemaform.com, 29/04/2017, http://www.rhemaforms.com/public/index.php?component=questionnairepane&formId=3&questions).

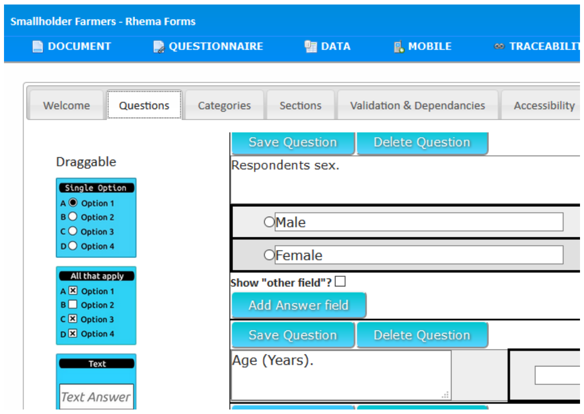


Figure 1: Questions design interface.

Server Side

The main aim of server side implementation was to update the database so as to reflect modifications made by the user on the client side. As such, similar to the client side logic, there were two implementations of server side code for manipulating questions. The first implementation was for saving an array of questions as received from the client. The other processed requests on a per question basis.

Multi Question Processing

Multi question processing is ideal for processing an array of questions received form the client. We used a traditional algorithm, shown in Figure 1, on the server to persist client submissions. The code compared the incoming array with corresponding records stored in the database. In case of any similarities between the two, the database would be updated appropriately. For example, if an element was found in the database but was missing in the array coming from the client, it would be deleted on the assumption that the user must have deleted it, hence its absence. In addition, every time this code was invoked objects had their positions updated to reflect any reordering that may have taken place on the client side. This is definitely costly but it is better than performing an if operation to check if position changed then performing an update. Finally, if any question was found in the incoming array, but not in the database, it was persisted. The items where persisted and their positions updated accordingly to reflect any changes performed on the client.

```
WHILE i < a[].length] DO
    deleteflag=true
    WHILE j < b[].length DO
        IF a[i]==b[j] DO
            deleteflag=false
        END IF
    END WHILE
    IF deleteflag==true DO
        remove a[i]
    END IF
END WHILE
WHILE j < b[].length DO
    insertflag=true
    WHILE i < a[].length DO
        IF b[j]==a[i] DO
            insertflag=false
        END IF
    END WHILE
    IF insertflag==true DO
        insert b[j]
    ELSE
        replace a[i]=b[j]
    END ELSE
END WHILE
```

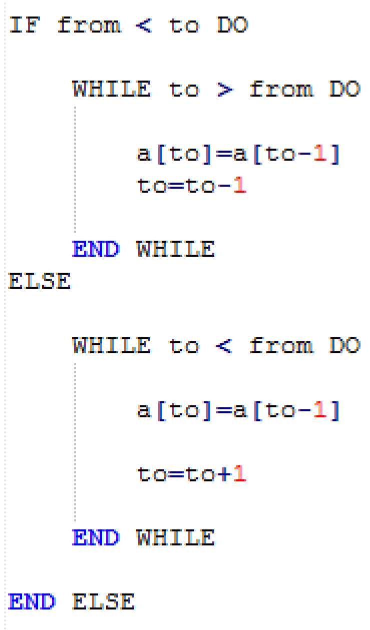
Figure 1: The traditional algorithm.

When synchronizing the client and server, the server side script fetches all corresponding items from the database and stores them in an array hereby referred to as **a[]**. In coming items received from the Json string are stored in array **b[]** . The first operation performed is to identify questions that a user may have deleted so that they can also be removed from the databased. As such for a every item in **a[]** a loop is run on array **b[]**. If a particular item is not found in array **b[]** it is deleted from array **a[]**.

Secondly, newly added questions are identified and inserted into the database using the second loop in which each item in array **b[]** is checked against array **a[]**. If a particular item is missing in array **a[]**, it is persisted to the database. Here we could have added element be **b[j]**, where j is a subscript, to array **a[]** as shown in the pseudo code in Figure 1.However, this would increase the size of the inner loop and consequently increase execution time. If an item is found in the array an update is performed with the expression a[j] = b[j]. As can be shown in Figure 1, the algorithm updates all the elements during its execution.

Single Question Processing

In contrast to the above mentioned concept, we present a single question submission approach that makes use of an array –shift algorithm. This approach separate delete operations from the CRUD process. However, update of a question's positions as well as content are still coupled together in that one can move an item from on position to another, edit its content then click the save button on that particular question. This question will asynchronously be submitted to the server where it will be processed using the algorithm presented in Figure 2.



```
IF from < to DO

    WHILE to > from DO

        a[to]=a[to-1]
        to=to-1

    END WHILE
ELSE

    WHILE to < from DO

        a[to]=a[to-1]

        to=to+1

    END WHILE

END ELSE
```

Figure 2: Array shift algorithm

The algorithm assumes that all items being sorted are already in the array. As such if a question being submitted does not exist, it first has to be inserted then the array is sorted. In this implementation we always had to append such an item to the end of the array and update its position to the position of the last element plus one. To update a question, the server – side script expects to receive the position of the question on the client side and it obtains the previous position by querying the database. In our algorithm the variable representing the previous position is referred to as **to** and the latter is referred to as **from**. As can be seen from the algorithm the only elements that get updated are those lying in the range including the previous and current position of the submitted question, are hereby being referred to as **to** and **from**.

IV. Test Results

In our test we measured sever side computation, by observing the amount of time taken to process a dataset submitted by the client to the server, on a 64-bit machine powered by an Intel(R) Core i5-2520M CPU running at 2.50GHz and having 4GB RAM. There are a number tools that offer a way of capturing response time as means of measuring computational costs. These include tools such as Mozilla's Firebug and Chromes Postman. We however, chose to use a code embedded script to measure time execution so as to exclude network related latency.

Two sets of tests were conducted, and response time was observed for both algorithms. In the first case seventy question where entered and execution time in milliseconds was recorded when a single question was dragged and dropped from position one to position two, position one to position three, position one to position four and so on until the last item was reached. As can be seen in Figure 3, the objective for this test was to observe the response time taken for n number of swaps for each algorithm.

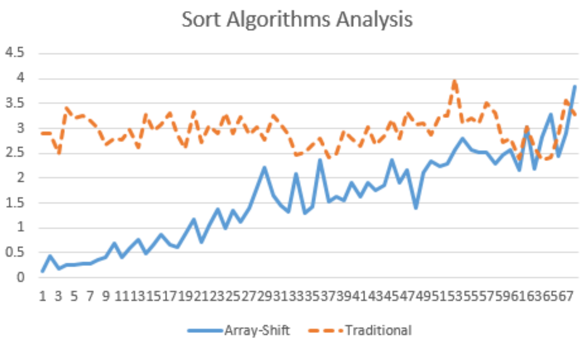


Figure 3: Comparing response time for varying swaps in a constant dataset.

The second objective was to vary the total number of items in the dataset and then move the first question to the bottom of the list and record execution time for both algorithms. The initial entry was for two questions, then three questions and so on until thirty-nine questions where observed. As shown in Figure 4 the results show a different perspective of the merging point for the two graphs observed in Figure 3.

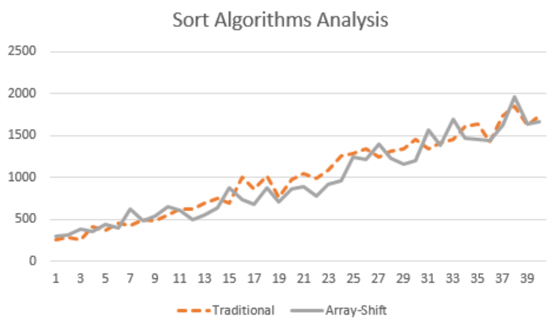


Figure 4: Comparing execution time for varying swaps in a varying datasets.

V. Discussion

The two algorithms in this experiments demonstrate how attaining usability and efficiency may be contrary one to another.

The traditional algorithm is biased towards ease of use by allowing the user, through drag and drop, to create, delete, update and swap questions in a single click. The save button can be clicked once for all changes and persist them to the web server which in turn updates the database accordingly. The approach has two performance issues: the first one has to do with a toll it has on bandwidth due to its indiscriminate submission of questions to the server. Most of the questions in a dataset may not have been modified on the client side. As much as this method may bring about a sense of high usability it actually cuts down on the benefits of partial page updates for which AJAX was developed. The other limitation of this approach is that it has a high computational cost as can be seen in Figure 3. The algorithm attempts to perform CRUD operations in one submission hence performing too many computations regardless of whether the user intended to perform such operations or not. The approach also falls short in terms of server side computational optimization due to the fact that all items in the database have to be updated in every submission. The algorithm works at worst case scenario regardless of the number of swaps.

On the contrary, Array-Shift algorithm takes an optimized approach in its usage of both bandwidth and server side computation. Submitting a question at a time as opposed to sending the whole dataset, optimizes on bandwidth usage. However, this may impact negatively on usability as the user is required to click a save button for each and every change made and he or she wishes to persist. It would be desirable to replace these save buttons on questions with automatic submission when a question is drop at a new position. The other way to perform automatic submission would be when a user finishes updating text in a text field by raising a mouse event. However, this would lead to unnecessary communication with the server which may result in wasted bandwidth. Therefore, a button on each question that can be clicked only when there was intentional action by the user to persist changes to the server was the most economical solution as far as both server side computation and bandwidth optimization is concerned. As can be seen from the test results in Figure 3 the graph for Array-Shift algorithm show significant performance gains. It was observer that the fewer the number of swaps the lower the response times. As the number of number of swaps nears the total number of items in the dataset, the poorer the performance. It can be seen that, when swaps are equal to the number of items in the datasets, response time is at its worst for the algorithm. At this point the two graphs merge. As shown in Figure 4 the second algorithm one is superior to the first and the two only have relatively equal performance when the number of swaps are equal to n-1, where n is size of the datasets.

VI. Conclusion

We observed that much as JQuery in collaboration with AJAX can enable the development of web based rich user interface applications that have performance benefits as opposed to traditional synchronous http based applications, there is still need to implement them using appropriate algorithms. By comparing the array-shift algorithm and a traditional no standard algorithm we have demonstrated that the usage of appropriate algorithms to implement asynchronous web applications can lead to improved performance. The performance gains in bandwidth and server –side computation optimization can be very useful for resource constrained environments such as developing countries. we have also observed that to develop high performance AJAX based web applications that have a rich user interface, some tradeoffs on the part of usability have to be. In our future works we are proposing to investigate algorithms that are not only biased towards performance, but are also able to sustain high software usability.

REFERENCES

[1] Md. Shahadat Hossain Khan,Mahbub Hasan, Che Kum Clement, "Barriers to the Introduction of ICT into Education In Developing Countries: The Example of Bangladesh," *International Journal of Instructio,* 2012.

[2] R. C. Sharma, "Barriers in Using Technology for Education in Developing Countries," Newark, New Jersey, USA, 11-13 Aug. 2003 .

[3] Jorge Alvarez, Peter Nuthall, "Adoption of computer based information systems:The case of dairy farmers in Canterbury, NZ, and Florida, Uruguay," *Computers and Electronics in Agriculture,* 2006.

[4] Jay Chen, David Hutchful, William Thies, Lackshminarayanan Subramanian, "Analyzing and Accelerating Web Access in a School in Peri-Urban India," *http://dl.acm.org/citation.cfm?id=1963358*, p. 11, 2011.

[5] Jingjing Li, Chunlin Peng, "JQuery - based Ajax General Interactive Architecture," *ieeexplore.ieee.org/document/6269466/,* 2012.

[6] G. Wang, "Improving Data Transmission in Web Applications via the Translation between XML and JSON," *Third International Conference on Comminications and Mobile Computing,* p. 4, 2011.

[7] R. Dhand, "Reducing Web Page Post Backs through jQuery Ajax Call in a Trust Based Framework," *Proceedings of the 2012 International Conference on Computing Sciences,* p. 3, 2012.

[8] C.-K. Shene, "An Analysis of Two In- Place Array RotationAlgorithms," *The Computer Journal,* 1997.

[9] Mwenge Mulenga, Luckson Simukonda, "www.rhemaforms.com," Kabwe, 29/04/2017.