JavaScript tools for online information retrieval

Article *in* Online Information Review · July 2006

DOI: 10.1108/14684520610686779 · Source: OAI

2 authors, including:

Ruwan Chinthaka Gamage
University of Colombo
18 PUBLICATIONS **22** CITATIONS

Refereed article received
26 January 2006
Revision approved for
publication 30 March 2006

JavaScript tools for online information retrieval

Ruwan Gamage
School of Information Management, Wuhan University, Wuhan, China and Library, University of Moratuwa, Moratuwa, Sri Lanka, and
Hui Dong
School of Information Management, Wuhan University, Wuhan, China

Abstract

Purpose – Efficiency of server side search engines is very low in cases of slow internet connections. Therefore, this study aims to examine use of client side search tools.

Design/methodology/approach – A previously introduced client side JavaScript search model was used. New data were obtained for response times against an array of different sized index files. A simple linear regression model was used to obtain the limitation of file size for the search tool. Response times for repeated searches were obtained for the client side search model and selected server side search tools.

Findings – It was found that the search model could be used only for a small-sized data set. Still, it was useful against server side search methods for repeated searches during a single session.

Research limitations/implications – Response time differs according to the network traffic, connection speed, and so on. Therefore, use of the search model is context-specific.

Originality/value – The model is easy to use and maintain. Therefore, organizations that wish to make their small data collections searchable on the web can use the model. The model is especially suitable for users with slow internet connections who experience very low efficiency in searching large server side databases. The paper introduces the model, solutions and technical aspects for practical execution.

Keywords Java, Search engines, Information retrieval, Response time

Paper type Research paper

1. Introduction

A script running on a client workstation might check the input user's submission to a web page. This is to make sure they entered all required data and appropriate data values. This can resolve local problems locally, without troubling the server for all minor issues. However, scripting has so many other uses, including advanced browse features and creating cookies on client machines. The server can use one type of cookies for tracking user actions on the client, and another type for "acting" as "search engines".

We are using a JavaScript search tool, one with an array of data elements to examine client side search tools' fitness to be a technology-in-demand for small databases. Lab testing was conducted, and response times were measured for displaying results case-by-case, with different sized data arrays.

The first part of the paper describes the motivation behind this study and its objectives, and some of the concepts used will be explained. The next section describes

Online Information Review
Vol. 30 No. 4, 2006
pp. 380-394

1468-4527
DOI 10.1108/14684520610686779

previous related attempts, and the search tool and results of experiments will be analyzed. Thereafter, a discussion and conclusions follow.

2. Motivation and objectives

It is common for people to spend more time in front of computers waiting until information appears, than actively extracting information. This is especially true in countries where internet access is comparatively slow. When it takes more time to receive results (response time), the search efficiency is low. Conversely, when there is a lower response time, the efficiency is high (Figure 1).

Client side search tools have the inherent feature of delivering the search results without consulting the server. If some progress can be achieved towards using JavaScript as an information retrieval solution, developing countries with slow internet connections will benefit. As for many, more time on the internet means more money spent. It is observed that users give up retrieval of web-based information because of the intolerable time lag for results.

Therefore, the main objective of this study is considering JavaScript for increasing search efficiency.

3. Concepts and tools

3.1 Client side search (CSS)

In server side search (SSS), the server handles all the requests. A busy server is likely to run out of memory when a large number of simultaneous services are requested. As interactive web access gained popularity, technologists developed new methods to process form inputs without starting a new copy of the servicing program for each browser input. Examples of these technologies for communicating with web servers include Java Servlets and Microsoft's ASP.NET; they allow a single copy of the servicing program to service multiple users without starting multiple instances of the program (Morrison *et al.*, 2002). Busy servers readily put these into use. Figure 2 shows the basic functioning of a client-server search engine.

Still, creating a short script is faster than creating a short compiled program. Also compiled programs demand advanced software from the user's terminal. Therefore, use of scripts is justified when a large program is not required for processing HTML form inputs (Morrison *et al.*, 2002). Scripting is a client side technology. A basic model of a client side program has been given in Figure 3.

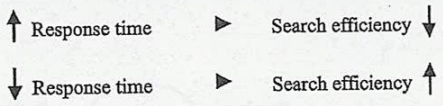

Figure 1.
Response time and search efficiency

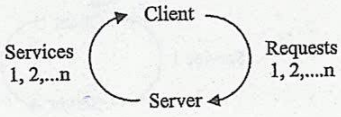

Figure 2.
Basic functioning of client-server search engine

Web pages associated with a compiled client side program run on the user's computer. A user's browser must be able to run the program file.

Client side processing is carried out on the client machine. Initially, the search script is requested from the server. Thereafter, for example, a JavaScript search tool writes cookies on the browser. These temporary files do the subsequent searching, avoiding the need for repeated requests from the server. The cookie is stored in the client's browser memory. This makes the client a host of the data set. The second search uses these cookies to give results. Succeeding searches also use the same information. The cookie's lifespan is only a search-session wide. If you close the computer or delete all the cookies, then you have to reconfigure the client for a new search.

In this type of client side script, source code written in such languages as JavaScript and VBScript is embedded in an HTML document. It is placed within delimiter tags (< script... > and </script >) to indicate to the user's browser that the text is code rather than web page text. If the user's browser is able to recognize and interpret the code, it is processed. Use of a client side scripting language depends on a user's operating system, browser platforms, and developer expertise. If the web pages are to be accessed by a variety of users over the internet, JavaScript is probably better than VBScript, as JavaScript is the only scripting language able to run on nearly all browsers (Morrison *et al.*, 2002).

3.2 Response time (lag time) – RT

Technically, response time refers to the amount of time it takes for a keyboard input to reach the application and return a response. Length of "response time" depends on various factors. Response time in a web-based search engine is expected to be proportional to the bandwidth, web traffic at the time, and performance of the PC. The higher the response time, the more embarrassing it is for the user. Previous research suggests that user productivity is dramatically reduced when response time is significantly high. Sterbenz (2001) states that further productivity gains are realized when the response time decreases to the range of 100 milliseconds. According to him, human factor studies have also indicated that consistent response time is better for users than response with a significant variance, since users alter their behavior based on response time at a relatively slow rate.

Nielsen (1997) confirms that a 0.1 second (100 ms) threshold is suitable, while a 1.0 second limit is acceptable. Within this limit a user's flow of thought will be uninterrupted. Ten seconds is the limit the user can focus their attention.

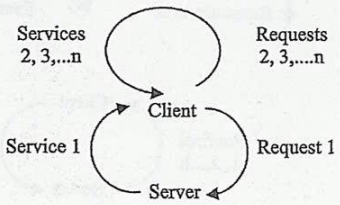

Figure 3.
Model of client side program

3.3 Zero response time
The ideal response time for a search is 0.00 seconds. No search tool has so far obtained this efficiency.

3.4 JSS and JavaScript array
There are many models for JSS, developed by many individuals. Here we are focusing on scripts with data arrays. The array contains data elements, or "objects" subjective for search. It is a listing of information arranged one after the other, with proper formatting, which is understandable by the script (see Appendix).

4. Literature review

Various attempts have been taken in the field of information retrieval to increase response time. Chan and Ueda (2000), Long (2002), and Quah *et al.* (2004) experimented with using cached objects, query slicing techniques, and web agents, respectively. All these are technologies associated with SSS. Academic research in CSS is rare. However, such applications are common in the internet for downloading – though, the usage in online applications is low.

JavaScript is the first choice in local search media, such as CD-ROMs and off-line databases. JSE (described later) is an example of a script using data arrays. This has also enjoyed a limited use as an internal web site search engine. Its function was to search within web sites. It was advised to use JSE with an array for less than 50 data elements (JSE Documentation, n.d.). Gamage (2006) used a basic test with few data sets to demonstrate the behavior of the same JavaScript search tool.

Advanced uses of JavaScript (not necessarily arrays) as an online information retrieval tool can also be found. WebSPIRS 4.0 by SilverPlatter is one such commercial application, which started using this technology for its search interface in the 1990s. ICDL, a recent project from the University of Maryland, USA, uses JavaScript for simple searches of children's books. It is also used for easy browsing of categories. Following is an overview of these two previous examples.

4.1 WebSPIRS
WebSPIRS 4.0 is a product by SilverPlatter (later absorbed by OVID). It is an advanced information delivery system that uses JavaScript for information retrieval (Jacsó, 2004). It was introduced in an era where most other information delivery systems were running on DOS or installable client software. WebSPIRS itself passed these "primal" stages. The version focused on was available for subscribers to SilverPlatter's online databases. Some institutions also mounted it on their Intranets. WebSPIRS presented features such as selecting search variables, index browsing, cross databases searching, and on the fly formatting of results.

4.2 International Children's Digital Library (ICDL)
ICDL (see www.icdlbooks.org) makes children's books available worldwide for free. Currently it has 914 children's books written in 34 different languages (Jacsó, 2004). Designed to support early literacy for children aged 5-10, it is a five-year research project of the University of Maryland's Human Computer Interaction Research Group. It offers two options for search through two parallel interfaces, one using a Java-based

zooming interface (termed "enhanced") and one using HTML and JavaScript (termed "basic"). The Java version supports conjunctive queries (i.e. read AND long books), but only works on certain web browsers. The HTML version works on any web browser, but does not support such queries (Hutchinson *et al.*, 2005). However, it worked with lower bandwidth and less powerful computers (Druin, 2005).

From the inception, the research team was changing features based on interaction research output. A critical issue that emerged was balancing access and innovation. When the ICDL was first launched in November 2002, only the "Enhanced" search option was available. Based on web log analyses, the team found that only 10 percent of all visitors coming to the ICDL web site actually used the library. This finding, along with other feedback from those who did not have hardware and software requirements, immediately convinced the team to focus on developing the "basic" interface for broad access. When ICDL Basic was launched in June 2003, the first five months of web logs showed that 50 percent of all visitors to the web site entered the library. This convinced the ICDL development team to reconsider the importance of developing tools for broad access. In addition, by having both versions, the team has been able to learn a great deal about the profile of users with regard to country of origin, categories of use etc., as opposed to dial-up (Druin, 2005).

5. The JavaScript search tool (JSS) on focus

5.1 JSE search

JSE Search is an open source JavaScript application downloadable from the Internet (JSE Documentation, n.d.).

JSE circumvents HTML's inability to pass a value from one page to another by using a session or non-persistent cookie. The cookie expires when the user's session ends. JSE consists of two HTML files (form.html and results.html) and two scripts (form.js and search.js). A data array is placed on the second script. Each element of the array takes the following format: TITLEURLDESCRIPTION$KEYWORDS. The field delimiter is the "dollar" ($) mark. Keywords are not displayed, while the title appears on top of the description with a link to the given URL (see the appendix for the script).

The first script writes cookies containing search words and then loads the result page. The second script reads the cookie, defines matches and generates search results. Results are set as a single-page numbered list with links to detailed pages if available. For users, searching is similar to "Google". A preceding minus character excludes a word, while phrases are supported within double-quotes. The existing model of JSE search is shown in Figure 4.

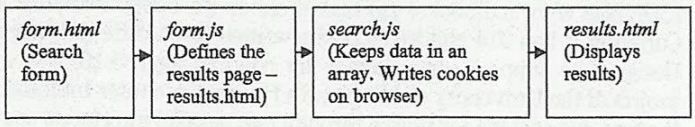

Figure 4.
Existing JSE search model

5.2 The response-time experiments

An experiment was conducted to understand the behavior of JSS response, with increasing the size of the data array. Actual web conditions were used by hosting a searchable directory on the internet. Response times were checked with data in the Sri Lankan Web Sites Database (www.srilankasupersearch.com), which contains directory type data: title, URL, description, and keywords for each data element (DE). An empirical 60 seconds RT limit was formulated, considering the readership, region on focus, and the nature of data.

Scripts with different numbers of data elements, thus having different script file sizes, were mounted in separate folders at http://search.arjees.com/testbed/jse All data was extracted from the same set of data for uniformity. A search was executed to test each JavaScript on the IE browser. The same word was searched (JSE) which had been inserted in a single data element (first DE) of each data set. Therefore, each search event retrieved only one result.

For each script, RT values were taken for the first search (action involved with the server) and for four subsequent searches (actions involved with the client only). After experimenting with each file size, cookies and temporary files were deleted. This was to ensure that no cookies were remaining in the client machine before the next experiment began. The PC with a 900 MHz processor, which we used to represent an average client, was connected to the institutional LAN. The server was located externally to the institution Intranet. It was attempted to control the variable network speed by taking all measurements while the throughput was between 50 kbps and 10 kbps. This throughput range lasted for only 85 minutes in the network. Values for 21 JavaScript file sizes could be collected within this period. The results are shown in Table I.

Another set of data was collected for comparison purposes. Table II presents response times obtained for selected popular search engines for five subsequent searches, and the response times obtained for JSE Search. The response time focused on for the first of JSE Searches is the empirical limit.

6. Data analysis and discussion

This is a study on evaluating the strengths and weaknesses of JavaScript as a search tool for small data sets. To examine the functionalities of JavaScript, JSE Search was selected. It is an open source, freeware, robust script that enables easy configuration.

This is an extension of a previous study by one of the authors (Gamage, 2006). In that study, only a few data sets were used. Results have not been compared with results from a server side search engine. The model introduced was fairly abbreviated. Those shortcomings have been avoided in the present study.

Analysis of the two tests conducted follows.

Results of the first test (Table I) were displayed in a bubble graph (Figure 5). The diameter of each bubble corresponds to the size of the particular script file. Results show the difference of SSS and CSS. The first search of JSE is corresponding to a SSS, because it has to download the necessary files and create cookies on the browser. All subsequent searches are client side processes. CSS has taken no time to give results.

Number of data elements in the JavaScript array	Size of JavaScript file – FS (kb)	Response times (s) First	Second	Third	Fourth	Fifth
0	2.3	–	–	–	–	–
1	2.5	10	≈ 0	≈ 0	≈ 0	≈ 0
10	4.8	10	≈ 0	≈ 0	≈ 0	≈ 0
20	7.8	11	≈ 0	≈ 0	≈ 0	≈ 0
30	10.9	15	≈ 0	≈ 0	≈ 0	≈ 0
40	14.2	11	≈ 0	≈ 0	≈ 0	≈ 0
50	17.2	5	≈ 0	≈ 0	≈ 0	≈ 0
60	20.1	3	≈ 0	≈ 0	≈ 0	≈ 0
70	23.7	8	≈ 0	≈ 0	≈ 0	≈ 0
80	27.3	7	≈ 0	≈ 0	≈ 0	≈ 0
90	30.5	8	≈ 0	≈ 0	≈ 0	≈ 0
100	33.3	8	≈ 0	≈ 0	≈ 0	≈ 0
150	49.7	15	≈ 0	≈ 0	≈ 0	≈ 0
200	66.8	29	≈ 0	≈ 0	≈ 0	≈ 0
250	86.4	15	≈ 0	≈ 0	≈ 0	≈ 0
300	104.0	24	≈ 0	≈ 0	≈ 0	≈ 0
350	115.0	23	≈ 0	≈ 0	≈ 0	≈ 0
400	128.0	28	≈ 0	≈ 0	≈ 0	≈ 0
450	142.0	35	≈ 0	≈ 0	≈ 0	≈ 0
500	164.0	30	≈ 0	≈ 0	≈ 0	≈ 0
600	189.0	45	≈ 0	≈ 0	≈ 0	≈ 0

Table I.
Increase of size of script and response time with number of data elements in the array for five subsequent searches

Name of search engine	URL	Response times (s) First	Second	Third	Fourth	Fifth	Total
Google	www.google.com	2	3	4	1	5	$\sum_1^5 G_{(RT)} = 15$
Yahoo	www.yahoo.com	10	20	30	12	50	$\sum_1^5 Y_{(RT)} = 122$
Alltheweb	www.alltheweb.com	15	240 +	10	10	5	$\sum_1^5 A_{(RT)} = 280+$
JSE Search	–	60	0	0	0	0	$\sum_1^5 G_{(RT)} = 60$

Table II.
Response times for five subsequent searches of popular search engines against JSE Search

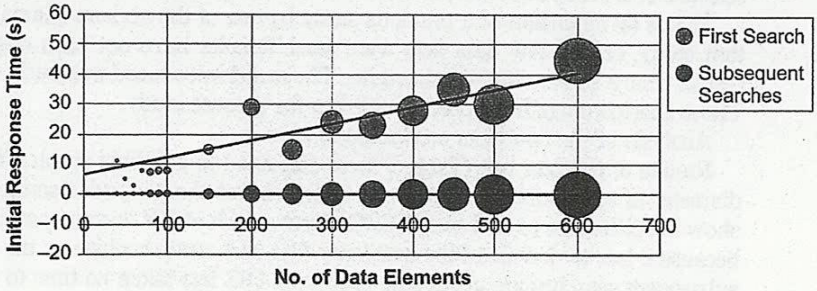

Figure 5.
Increase of size of script and response time with number of data elements in the array for five successive searches

Simple linear regression analysis was carried out between RT (dependant variable) and FS (independent variable). According to the analysis, the adjusted R^2 value is 0.812. That means approximately 81 percent of the variation of response time is affected by the script file size. The balance is due to random variables. The calculated "f" value is 82.849, and tabulated "f" value is 4.40 at 95 percent significance level. Therefore, the experiment is highly significant. From the data, it is 95 percent certain that an increase of file size by 1 kb will make the response time increase in the range 0.136-0.218. The limit of response time was kept at 60 seconds, as described in section 5.2.

According to the analysis, β_0 and β_1 values are 0 and 0.177 respectively. Therefore, the prediction equation can be interpreted as:

$$\hat{Y} = \hat{\beta}_{0(RT)} + \hat{\beta}_{1(RT)}X \tag{1}$$

$$Y = 0.177\,X \tag{2}$$

Y is replaced by the limit of RT (60 sec). X is the predicted FS limit.

Therefore $X = 60/0.177 \approx 340$ kb.

The obtained file size limit (X) is approximately 340 kb.

A similar analysis was carried out to measure the corresponding number of data elements for the given FS:

$$\hat{Y} = \hat{\beta}_{0(FS)} + \hat{\beta}_{1(FS)}X \tag{3}$$

Let the number of data elements be zero – according to the data obtained, the file size is 2.3 kb. Therefore:

$$\hat{\beta}_{0(FS)} = 2.3$$

According to the analysis:

$$\hat{\beta}_{1(FS)} = 0.318$$

Therefore, the prediction equation can be interpreted as:

$$Y = 2.3 + 0.318\,X \tag{4}$$

To obtain the number of data elements at the FS limit (for this set of data), substitute the FS limit to equation (4). The result is (X) approximately 1,062.

Therefore, we can predict that for this set of data, the maximum number of data elements we can host is 1,062 to search within a RT of 60 sec.

Test 2 demonstrates that during the process of repeated searching, each SSS also requires a time for subsequent searches. It is obvious that the cumulated RT in each case becomes more than the RT of JSE at a particular junction. This can be interpreted as:

$$\sum_1^p G_{(RT)}, \sum_1^q Y_{(RT)}, \sum_1^r A_{(RT)} > \sum_1^{p,q,r} J_{(RT)}.$$

The summation values represent total RT of Google, Yahoo, Alltheweb, and JSE in search numbers p, q, r respectively.

Therefore, JSE is more efficient in repeated searching than a server side search engine for a small number of data objects not exceeding a file size of 340 kb. These data are acceptable for the particular range of internet speed (throughput 50-10 kbps).

6.1 Discussion on results obtained

The experiment was designed to examine the behavior of JavaScript in managing databases. The network and terminal environment conditions represented a typical internet connection in a developing country. The increase of response time with the increase of number of data elements was tested. Results were presented in the form of a bubble graph, as it clearly displayed the change of RT with the size of the script.

In this case, it shows that the RT is proportional to the size of the script. Before the experiment we decided on an empirical limiting value for RT of 60 seconds. This was based on the target population (general public), place (Sri Lanka), and average internet connection speed (low) (Shrestha and Amarasinghe, 2001). Therefore, the study shows that in this particular case, around 1,062 data elements is the maximum that can be hosted.

From the results it is evident that JavaScript cannot be used for making search tools for long lists of data sets. It also demonstrates its inability to handle large files of book data. Therefore, both book catalogues and heavy databases are not the best candidates for JSS. However, the results demonstrate that the subsequent searches showed it is really fast in displaying results – a speed that no server side technology can match. Also, it is obvious from the study that the search tool is truly efficient for repeated searching of small data sets.

It should be noted that the second search in Alltheweb had the user wait for a long time. This may be due to an abnormality in the network traffic or some other unknown factor. However situations like this, and instances of compete breakdown of servers are common in server side searching. This highlights the suitability of script-based search for repetitive searches. This also satisfies our primary objective – having good response times.

6.2 Limitations of the study

Search engine studies carry out tests such as the use of Boolean logic, truncation, field search, recall etc., for examining the efficiency and effectiveness of a tool. However, we have concentrated on the "response time for word search" only. Also, the experiment was based on directory entries, rather than comprehensive catalogue data. Therefore, we cannot generalize the maximum number of data elements the search tool can handle within a justifiable RT.

Test conditions imitate a typical network and PC environment, and every effort has been taken to control every variable other than those tested. But it should be noted that the internet speed is constantly changing in a network, depending on network traffic. Processor speed is also changing from PC to PC.

To demonstrate repetitive searches in a server side search, we used huge databases utilizing different technologies. This may not represent an average server side search tool with a small number of data elements in its database.

6.3 Advantages of CSS

CSS is simple, and does not require prerequisites of hardware or special software for operation or installation. Therefore anyone can host a CSS even on a free web server. Because free servers limit server activity requested by client web sites, using CSS applications would be an advantage for the free-server customer. Whatever the server, server traffic will be at its minimum, because most activities are carried out in client machines. For the administrator, debugging is easy. Having all files necessary in a single folder on a local machine, any problem regarding the source and data can be resolved. In the case of JSS, only four lightweight files are needed.

Users will find recall good because each data set is small. High-speed search results and avoiding possible server breakdowns are other advantages. Users with inefficient Internet connections find JavaScript tools attractive. ICDL research findings support this attitude. Due to lack of popularity, they are phasing out the "enhanced" version of ICDL, which uses Java (International Children's Digital Library, n.d.). Instead they are focused more on developing the JavaScript version.

6.4 Disadvantages

The size of the data set is limited. Therefore the user has to search in different categories if the data set is larger than the amount it can hold. Response time for configuration is higher than the approved times given in section 3.2.

Because of the limitation on the size of data array, it has limited uses. It is possible to search only for metadata – not full-text or abstract. Anyone can download the script because the real URL of the file is given in the form.html page. Therefore, only public data can be hosted. However, links to abstract and full-text (where applicable) can be directed to server side techniques, which have more security. Users are sometimes skeptical in allowing JavaScript to run on machines, as there is concern about security and spying in script-enabled browsers. If a user's PC has not enabled cookies, the script cannot run, thus ignores the request. The display format is poor, and search controls (field search, search among databases etc.) are less.

From the viewpoint of an administrator, repeated off-line updating of the database is a problem. Therefore having real-time updates is not a reality, because as it is dependent on human work, it may or may not be updated regularly. Therefore the user cannot trust the available data.

6.5 Improvements

To satisfy the need to harness the wonders of scripting, while keeping users in touch with the search system, a new model was introduced. It is termed the two-tier JavaScript search model. Inclusion of data from a multi-field format, automation of writing the script, and the introduction of categorized searches are other improvements introduced. These are simple, but effective mechanisms in terms of hosting CSS.

6.5.1 Two-tier JavaScript search model (TTJSM). As explained above, using a JavaScript for searching means there is a delay between the first search request and displaying the results. Although the delay in the first search is high, the response times of subsequent searches become virtually zero. Therefore, there should be a method to negotiate with the user until the first search is carried out. The strategy used for this is

to first allow the user to do a configuration before carrying out the actual search. This configuration is actually a pseudo search (see Figure 6).

In order to achieve the pseudo search, one layer of action (two files) was added to the existing model (Figure 4). These are index.html (pseudo search page with a configuration button) and the form_f,js, which is the false form.js script. The button in the first HTML page sends a search command to the server. This initiates the procedure of writing cookies in a user's browser. After it is done, the user will be redirected to the real form HTML page with a search box. The whole process is called "configuration".

TTJSM is simple in structure, and easy to configure. It also reveals reasons for the delay to the user. Therefore there is no hiding of facts. If the user agrees to the deal because of the bonus of zero-response times for subsequent searches, he can stay and continue. Otherwise he can switch to a server side search tool.

Because the user is informed beforehand, they are not dejected by the long time taken for the first response.

However, there is a possibility that the user can leave the whole system, because explanations are lengthy and configuration is not a familiar practice in search engines. Sacrificing the first search attempt is another issue from the side of the user.

6.5.2 Enabling multiple fields inside display format and mechanizing script writing. The only file with dynamic data of the set of JSE files is search.js script. Therefore, if the creation of the script can be automated, it will be easier for the average technologist who manages the web catalogue.

Each data element in JSE has the format: TITLEURLDESCRIPTION$ KEYWORDS. Each variable is separated by a 'dollar' ($) mark. Hence the script we are focusing on supports the inclusion of title, URL, description and keywords of the data element. Some kinds of data elements, for example, a set of library books, contain more fields such as author, publisher, place etc. Therefore, we examined a particular bibliographic data set for inclusion within the same format. Formatted text with more fields could fit in the description area, without harming the functionality, or without changing the script. Then the script creation process was mechanized using a commonly used library database program.

We used CDS/ISIS 1.5 for Windows for mechanizing the writing of script. It is a freeware distributed by UNESCO as a database tool. Libraries in developing countries use this for developing their databases in electronic form. Although current usage records are not available, it is still a popular and powerful library software, and is taught in most library schools. Therefore, many small libraries have added or are

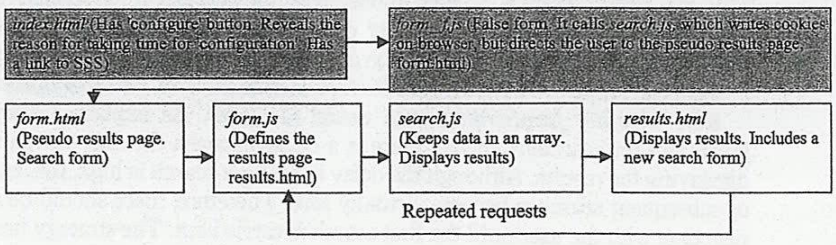

Figure 6.
Proposed TTJSM model

planning to add their book data into CDS/ISIS databases. Due to common usage, and many having already entered bibliographic data in CDS/ISIS, it is a plus point for using the program for mechanizing script writing. The idea is to print data array into a text file. This can be inserted into the search.js template.

The sample data set that we chose is the sample "Conference Proceedings Database" available in CDS/ISIS. We created a print format suitable for displaying results in a window. Both CDS/ISIS formatting language and HTML were used (Figure 7).

Data is displayed as follows. Additional charters can be seen according to the conventions used when entering data. The display format is fully adjustable according to an administrator's wish (Figure 8).

6.5.3 Category search. Because JSE can handle only a small number of results, comparatively large databases can be divided into sub-sets. This paper does not introduce a possible modification for searching all categories at one stretch. However, each category can be made a subject for a main search as shown in Figure 9. It is a demonstration of DDC classification in a category search.

6.6 Uses

It is true that users with high internet connection speeds, or those who use applications within Intranets enjoy the real flavor of CSS. However, those with low bandwidth may also need some applications when the information is critical, and repeatedly on use. Dictionary and thesaurus listings are one example, as is a catalogue or a directory. For example, researchers may need book data in libraries, or data on artifacts in a museum. A doctor may need information on essential medicines – information that can save a

```
's[',mfn(1),'] = "$$',V12,v24,,([([V76^Z|: |,V76^*|)
|)'<br><b>Author/s</b>:' ,V70+|; |'<br>',V25,V26,V30'<br>',|([V44|)
 |,V50,| '<br>'|V71| |V72| |V74'<br>',V69'$";'##
```

Figure 7.
CDS/ISIS print format for writing the data array

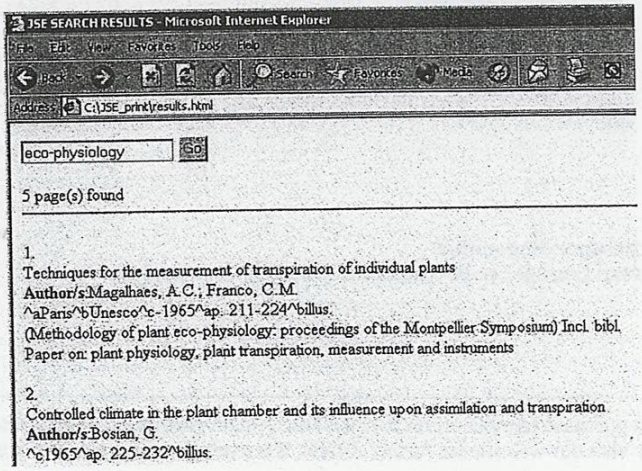

Figure 8.
Display of results in multiple fielded data elements

Appendix

form.js

```
// ---------- script properties ----------
var results_location = "results.html";
// ---------- end of script properties ----------
function search_form(jse_Form) {
        if (jse_Form.d.value.length > 0) {
                document.cookie = "d=" + escape(jse_Form.d.value);
                window.location = results_location;
        }
}
```

search.js

```
// ---------- script properties ----------
var include_num = 1;
var bold = 0;
// ---------- sites ----------

var s = new Array();

s[0] = "Please insert the data array below."
```

Header

```
s[1] = "Autosrilanka.com^http://www.autosrilanka.com/^<blockquote><b>http://www.autosrilanka.com/</b> >>>
Free advertisements (vehicles)</blockquote>^Autosrilanka Advertising free advertisements promotion vehicles
auto cars vans automobiles publicity promotions marketing sales";
s[2] = "EeZee2.com^http://www.eezee2.com/^<blockquote><b>http://www.eezee2.com/</b> >>> Free Classified
Advertisements.</blockquote>^EeZee2.com Advertising publicity promotions marketing sales";


// ---------- sites continue within this array----------

// ---------- end of script properties and sites ----------

var cookies = document.cookie;
var p = cookies.indexOf("d=");

// ---------- script continues ----------
--------
--------
--------
end.
```

Footer

Figure A1.
Form.js and search.js (part) JavaScript files

Corresponding author
Ruwan Gamage can be contacted at: ruwan@lib.mrt.ac.lk