# JavaScript Tools for Online Information Retrieval

Ruwan Gamage[1,2] , Hui Dong[1]
[1]School of Information Management, Wuhan University, China
[2]Library, University of Moratuwa, Sri Lanka.
ruwan@lib.mrt.ac.lk

**Abstract.** JavaScript has a comparatively long history as an online information retrieval tool. During the last decade SilverPlatter's popular WebSPIRS 4.0 started using JavaScript for its search functions. International Children's Digital Library is a current system that applies JavaScript for category based information retrieval. However, JavaScript capabilities for quick browsing and searching small collections is under utilized in light of advanced server-side technologies. Focussing on search engines using data arrays in scripts, this paper tries to justify one possible reason behind this - high response time for the starting search. To cope up with the situation, the paper introduces a model for interface design. Also it reveals that the script search is superior to server side techniques in terms to response time, when the user's session is several searches long.

Key words: JavaScript, Search Engine, Information Retrieval, Search Model, Response Time, Google

## 1. Introduction

A script running on a client workstation might check the input users submit 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. Server can use one type of cookies for tracking user actions on the client, and another type for *acting as search engines*.

Here 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. Conducted lab testing. Measured response times for displaying results case by case, with different sized data arrays.

The first part of the paper describes motivation behind this study, and its objectives. Then it will explain some of the concepts used inside. Next section describes previous related attempts. Then the search tool and results of experiments will be analyzed. Thereafter, a discussion and the conclusions follow.

## 2. Motivation & Objectives:

It is a common situation that people spend more time before computers waiting until information appears, than actively extracting information. Especially this is true in countries where the Internet access is comparatively slow. Higher the time to receive results (response time), search efficiency is low. Conversely, lower the response time, the efficiency is high (Fig. 1).

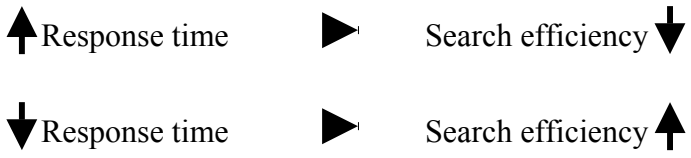



**Fig. 1 Search efficiency and response time**

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

Therefore the main objective of this study was considering the ability of JavaScript for increasing search efficiency.

## 3. Concepts & 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 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, 2002). Busy servers readily put these into use. Fig. 2 shows the basic functioning of a client-server search engine.

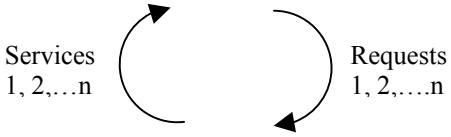



**Fig. 2 Server side search model**

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, 2002). Scripting is a client-side technology. Basic model of a client side program has been given in Fig. 3.

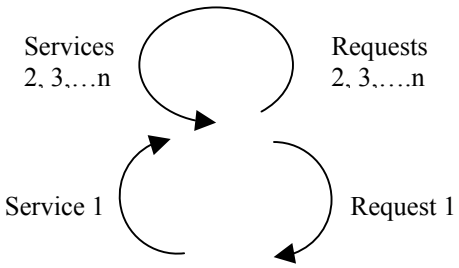



**Fig. 3 Client side search model**

Web pages associated with a compiled client-side program run on the user's computer itself. 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 client's browser's memory. This makes the client itself a host of the data set. The second search uses these cookies for giving results. Succeeding searches also use the same information. The cookie's lifetime span 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 scripts, 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 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, 2002).

### 3.2 Response Time (Lag Time) - RT

Technically, response time refers to the amount of time it takes for an input from a keyboard to reach the application and a response returned. Length of 'response times' 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. Higher the response time, it is more embarrassing 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 factors 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 0.1 second (100 ms) threshold is suitable while 1.0 second limit is acceptable. Within this limit users' flow of thought will be uninterrupted. Ten seconds is the limit the user can focus his attention.

### 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 focussing 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 1).

## 4. Literature Review

Various attempts have been taken in the field of information retrieval to increase the response time. Chan and Ueda (2000), Long (2002), and Quah (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 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 for a script using data arrays. This has also enjoyed a limited use as a web site internal search engine. Its function was to search within web sites. JSE was advised to use with an array for less than 50 data elements (JSE, 2005). 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 90's (Jacsó, 1998). ICDL (2006), a recent project of University of Maryland, USA, uses JavaScript for simple search 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 of SilverPlatter (later absorbed by OVID). It is an advanced information delivery system that uses JavaScript for information retrieval (Jacso, 1998). 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 on focus was available for subscribers to SilverPlatter on-line databases. Also some Institutions 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 http://www.icdlbooks.org) makes children's books available worldwide for free. Currently it has 914 children's books written in 34 different languages (ICDL, 2006). Designed to support early literacy for children aged 5-10, it is a 5-year research project of University of Maryland 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 & JavaScript (termed 'basic'). The Java version supports conjunctive queries (i.e. red 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, 2005). But 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 Weblog 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 didn't have hardware and software requirements, convinced the team immediately 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 who use what 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 Internet (JSE, 2005).

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* & *results.html*) and two scripts (*form.js* & *search.js*). Data array is placed on the second script. Each element of the array takes the following format; TITLE$URL$DESCRIPTION$KEYWORDS. The field delimiter of choice is 'dollar' ($) mark (changeable). Keywords are not displayed while the title appears on top of the description with a link to the given URL. See appendix 1 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 set is 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 follows.

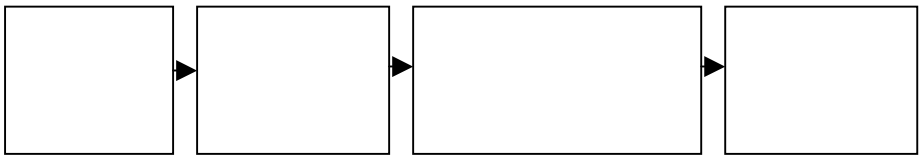


**Fig 4. Existing *JSE Search* model.**

### 5.2 The Response-Time Experiments

Conducted an experiment to get an idea on the behavior of JSS response with increasing size of the data array. Used actual web conditions by hosting a searchable directory on Internet. Response times were checked with data in the Sri Lankan Web Sites Database (www.srilankasupersearch.com). It contains directory type data; Title, URL, Description, and Keywords for each data element (DE). Formulated an empirical 60 seconds RT limit 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 were extracted from the same set of data for uniformity. Executed a search to test each JavaScript, on IE browser. Searched for the same word - 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 4 subsequent searches (actions involved with the client only). After experimenting with each file size, cookies, and temporary files were deleted. This was to make sure that no cookies were remaining in the client machine before the next experiment starts. 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 exterior to the institution Intranet. Tried to control the variable of network speed by taking all measurements while the throughput was between 50 kbps and 10

kbps. This throughput range lasted only for 85 minutes in the network. Values for twenty-one JavaScript file sizes could be collected within this period. The results follow.

**Table 1. Increase of size of script and response time with number of data elements in the array, for five subsequent searches**

| Number of Data Elements in the JavaScript Array | Size of JavaScript File - FS (kb) | Response Times -RT (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 |

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

**Table 2 : Response times for five subsequent searches of some popular search engines against *JSE Search***

| 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 | $\sum_{1}^{5} G_{(RT)} = 60$ |

## 6. Data Analysis and Discussion

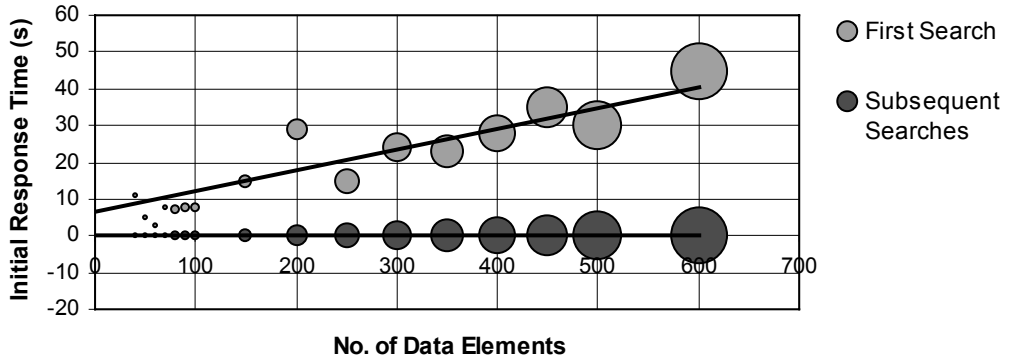
This is a study on evaluating the strengths and weaknesses of JavaScript as a search tool for small data sets. To examine the functionalities of JavaScript, JSE_Search was selected. It is an open source, freeware, robust script that enables easy configuration.

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 1) were displayed in a bubble graph (Graph 1). 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 browser. All subsequent searches are client side processes. CSS has taken no time to give results.

**Graph 1. 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, adjusted $R^2$ value is 0.812. That means approximately 81% of the variation of the 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% significance level. Therefore the experiment is highly significant. From the data it is 95% certain that an increase of file size by 1 kb, make the response time increase in the range 0.136 - 0.218 s. Kept the limit of response time at 60s, as described in section 5.2.

(1) is the prediction equation.

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

According to the analysis, $\beta_0$ and $\beta_1$ values are 0 and 0.177 respectively. Therefore the prediction equation can be replaced by (2)

Y = 0.177 X ---------- (2)

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

Therefore X = 60/0.177 ≈ 340 kb

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

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

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

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 (4).
Y = 2.3+0.318X ----------(4)

To obtain the number of data elements at the FS limit (for this set of data), substituted the FS limit to equation 4. The result is (X) is approximately 1062.

Therefore we can predict that for this set of data, the maximum number of data elements we can host is 1062 to search within a RT of 60s.

Test 2 demonstrates that during the process of repeated searching, each SSS requires a time for subsequent searches also. 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. This data is 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. Increase of response time with the increase of number of data elements was tested. Results were presented in the form of a bubble graph, because 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; 60 seconds. This was based on the target population - general public, place - Sri Lanka, and average Internet connection speed - low (Shrestha, 2001). Therefore the study shows that in this particular case, it is around 1062 data elements is the maximum that can be hosted.

From the results it is evident that JavaScript can not 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, 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 made the user waiting 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 search. This also satisfies our primary objective of having good response times.

**6.2 Limitations of the study**

Search engine studies carry out more tests such as use of Boolean logic, truncation, field search, recall etc. for examining 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 can't 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 noticed 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 server-side search, we used huge databases using 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 doesn't require pre requisites 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 it's minimum, because most activities are carried out in client machines. For the administrator, debugging is easy. Having all files necessary on a single folder in a local machine, any problem regarding the source and data can be resolved. In the case with JSS, only four lightweight files are needed.

User 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 (ICADL, 2006). Instead they concern more on developing the JavaScript version.

### 6.4 Disadvantages

Size of the data set is limited. Therefore 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 that 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. User is sometimes skeptical in allowing JavaScript to run on machines, because there is a concern on security and spying in scripts enabled browsers. If users' 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 administrator, repeated off-line updating of the database is a trouble. Therefore having real-time updates is not a reality. Because it is dependent on human work, it may or may not be updated regularly. Therefore the user cannot keep trust on the available data.

### 6.5 Improvements

To satisfy the need of harnessing benefits of scripting while keeping users intact with the search system, a new model was introduced. It is termed as the Two-Tier JavaScript Search Model. Inclusion of data from a multi-field format, automation of writing the script, and introduction of categorized search 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 of results. Though 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 here for this is first allowing the user to do a configuration before doing the actual search. This configuration is actually a pseudo search (See fig.5).

In order to achieve the pseudo search, one layer of action (2 files) was added to the existing model (Fig. 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 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'.

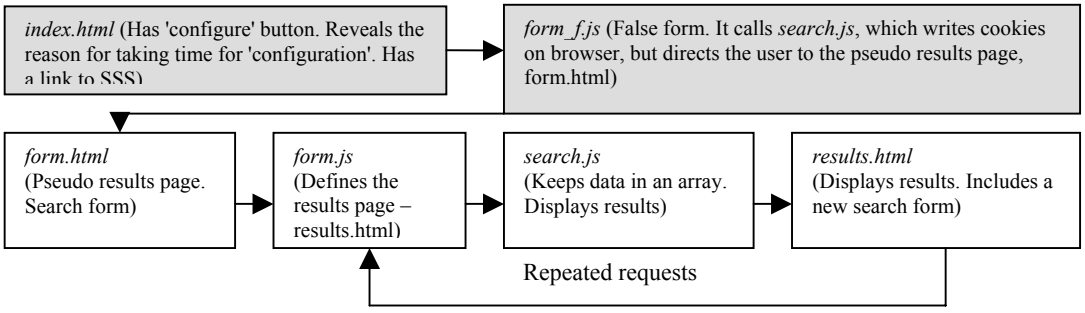



**Fig 5. Proposed TTJSM model.**

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

Because the user is informed beforehand, user is 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.j*s 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 TITLE**$**URL**$**DESCRIPTION**$**KEYWORDS. Each variable separated by a 'dollar' ($) mark. Hence the script which we are focussing our attention support 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 making their databases in electronic form. Though current usage records are not available, it is still popular & a powerful library software and is taught in most library schools. Therefore many small libraries have added or planning to add their book data into CDS/ISIS databases. Common usage and many having already entered bibliographic data in CDS/ISIS 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.

```
'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'$";'##
```

**Fig. 6 CDS/ISIS Print Format for Writing the Data Array**

Data is displayed as follows. Additional charters can be seen according to the conventions used in entering data. The display format is fully adjustable according to administrators' wish.

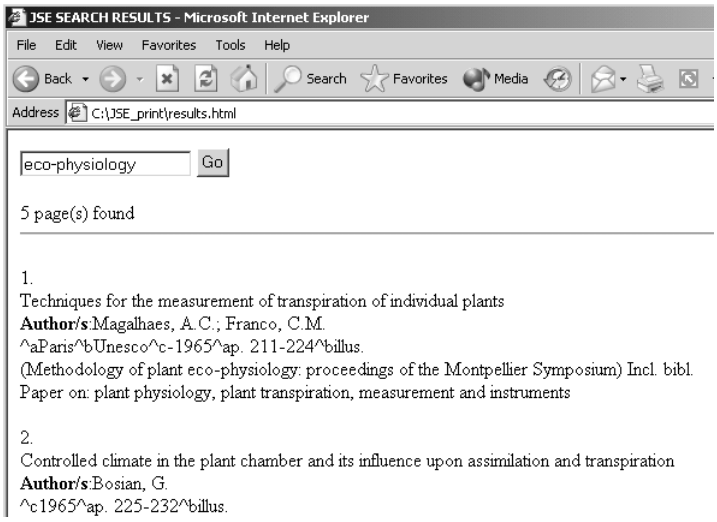


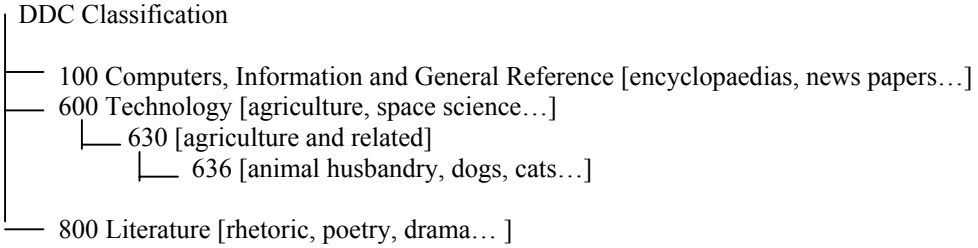

**Fig. 7 Display of results in multiple fielded data elements.**

#### 6.5.3 Category Search

Because JSE can handle only a small umber of results, comparatively large databases can be divided into sub sets. This paper doesn't introduce a possible modification for searching of all categories at one stretch. However, each category can be made a subject for a main search as shown in Chart 1. It is a demonstration of DDC classification in category search.

**Chart 1. Description of category search**

***Organization of Topics:***




***Data Array for Category search:***

(header)
"s=[1] 100 Computers $comp.html $Description 1 $ encyclopaedias, news papers, …
"s=[2] 200 Technology $tech.html $Description 2 $ agriculture, space science, …
"s=[3] 636 Animal husbandry $animal.html $ Description 3 $ animal husbandry, dogs cats, …
"s=[4] 800 Literature $lit.html $Description 4 $ rhetoric, poetry, drama, …
---
---
---
(footer)

***Example for a Search:***
Search for the keyword 'dogs'.

***Results:***
Search within…
1. 636 Animal husbandry (hyperlink to animal.html)
   Description 3

### 6.6 Uses

It is true that the real flavor of CSS is enjoyed by users with high Internet connection speeds or those who use applications within Intranets. However, those with low bandwidth also may need some applications when the information is critical, and repeatedly on use. Dictionary & Thesaurus kind of listing is one example. It can also be 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, which can save a life. A lawyer who practices International Law may need information from recent cases from overseas. Most of the above information is both critical, and urgent. It is also needed repeatedly during a single session.

Other information which may or may not be critical; such as product catalogues, and web directories also qualify for possible hosting as CSS.

### 6.7 Future

More studies, specially user studies should be carried out on the acceptance of the model. Code can be changed to give more control on search. Use of frames and other technologies should be examined for simultaneous configuration of several categories.

## 7 Conclusion

Using CSS for imitating databases for small collections can improve customer satisfaction by giving speedy access to search results. It is a matter of fact that the configuration time is high in low bandwidth connections. In that sense it is viable to state that JSS engines are suitable for Intranets than the Internet. For example universities officering information products to their own students and staff can reap benefits of this technology. Demanding the user to search in several search-spheres is apparently a backward option, but it is worthwhile compared with virtually zero response time in subsequent searches. This would enable catalogues and directories already on WWW to improve their search options. Also, these models will be helpful to immediately host web OPACs for those who have not yet done so.

This would enable high-speed basic search in a very small catalogue, usually with a script with data of less than 340kb. Researchers and Professionals, who need to search repeatedly for information during the course of the day, would be the likely users.

One should not always assume that using a JSS engine means some amount of delay. If the number of data objects is very low, or the Internet speed is very high, the system may configure itself very quickly.

CSS can be an option in the main interface for the user to select.

JavaScript is useful for navigation between information categories. ICDL applied this for good use.

## References:


1. Chan E. and Ueda K. (2000). Efficient Query Result Retrieval Over the Web. In Proceedings of International Conference on Parallel and Distributed Systems. IEEE Computer Society.
2. Druin,A. (2005). What Children Can Teach Us: Developing Digital Libraries for Children with Children. The Library Quarterly. 75/1. pp. 20-43
3. Gamage, Ruwan (2006). Where the Speed Matters... Zero-Response-Time Search Engine for Small Collections. Ed. Fox, E.A., Neuhold, E.J., Premsmit, P. et al. Proceedings of 8th International Conference on Asian Digital Libraries. Lecture Notes in Computer Science. Vol. 3815. January, 2006. Springer Berlin/Heidelberg. pp. 224 - 231.
4. Hutchinson, H.B., Rose, A., Bederson, B.B. et al (2005). The International Children's Digital Library: A Case Study in Designing for a Multi-Lingual, Multi-Cultural, Multi-Generational Audience. Information Technology and Libraries. American Library Association, 24/1, pp. 4-12.
5. ICDL (2006). International Children's' Digital Library. www.icdlbooks.org Visited on 03.03.2006
6. Jacsó, P. (1998) WebSPIRS 4.0: JavaScript Search Software. Library Software Review 17(4). pp. 244-269.
7. JSE (2005). Downloaded from http://www.Javascriptkit.com/script/script2/jse/jse10a.zip on 30.05.2005.
8. Long, B. A (2002). Design Pattern for Efficient Retrieval of Large Data Sets from Remote Data Sources in on the Move to Meaningful Internet Systems, LNCS 2519. pp.650–660.
9. Morrison, M. Morrison, J. & Keys, A (2002). Integrating Web Sites and Databases. Communications of the ACM. September 2002/Vol. 45, No. 9.pp 81-86
10. Nielsen, J. (1997). The need for speed at http://www.useit.com/alertbox/9703a.html. Retrieved, June 2005.
11. Quah, J.T.S., Chen, Y.M., Leow, W.C.H. (2004). Networking E-Learning Hosts Using Mobile Agents. Ed. Mohammadian, M. Intelligent Agents for Data Mining and Information Retrieval. Idea Group Inc., Hershey (2004). 263-293.
12. Shrestha, G., Amarasinghe, S. (2001). Perspectives on the Use of the Internet in Sri Lanka. LCS Technical Report TR-815. Massachusetts Institute of Technology, USA. 2001. Available at http://www.cag.lcs.mit.edu/~saman/papers/SLReport-1-01.pdf Visited on 03.03.2005.
13. Sterbenz, J.P.G. (2001) High-Speed Networking: A Systematic Approach to High-Bandwidth Low-Latency Communication. John Wiley & Sons, Incorporated, New York. 441-442.

## Appendix1: form.js and search.js (part) JavaScript files.

***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.autosrilan
ka.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**