Data Processing Across the Internet: A Model for Design

William J. Montelpare, Ph.D.[1] Moira N. McPherson, Ph.D.[2]

[1]Associate Professor and Director, School of Kinesiology, Lakehead University;

[2] Associate Professor, School of Kinesiology, Lakehead University

Corresponding author: William J. Montelpare, Ph.D.; Associate Professor and Director; School of Kinesiology; Lakehead University; 955 Oliver Road; Thunder Bay, Ontario P7B 5E1; phone: 807. 343.8481; fax: 807.343.8944; email: WMONTELP@FLASH.LAKEHEADU.CA

Abstract

The significance of the Internet, as a tool for health researchers has yet to be determined. Although search engines have been developed to use the Internet in health database information retrieval, the current practice is for health researchers to use the Internet as a billboard to post facts and information. This paper describes a model for design, which has been used to extend the Internet's capabilities in health research beyond the elementary posting of facts through to the collection of data. The benefits of combining a scripting language with a common gateway interfacing software are exemplified through an application in academic research.

Introduction

The utility of the Internet provides researchers with a tool that can be used to present information in several forms at a variety of venues (Maddux and La Mont-Johnson, 1997). Health researchers, health educators, and practitioners delivering health services, currently use the Internet to discuss topics, describe outcomes, and advertise products. A simple Internet search, using the widely available search-engine Alta-Vista©, on "health" as a single keyword, recognized over one million health based web sites.

Yet the Internet at its basic level, is merely a repository for "posting" facts and information to be retrieved by passive "browsers". Web authors can, however, move out of the billboard approach and create a sense of interactivity with a dynamic presentation of information, by incorporating the scripting features of an interpreted language, or the more complex interface features of compiled programs.

As the Internet is not hardware dependent (Collis, 1996), a researcher can use the Internet to present information in the form of web pages, and collect specific information from web page visitors, independent of individual computer platforms. Researchers can create project specific applications that are presented to a target audience, without regard for operating systems or the type of computer that will receive the information. Through a strategic plan using most commercially available "web browsers", researchers can enhance their ability to develop knowledge from the acquisition of end-user information (Stemler, 1997). Researchers can evolve from the level of posting and browsing information, to the level of data collection.

Yet few researchers have described, in theoretical terms, either the nuances of, or techniques involved in, collecting and processing data across the Internet. The purpose of this paper is to demonstrate the application of a theoretical model in the design of Internet-based data collection for health research. Most important, the techniques presented here use standard formats and procedures which are easily adaptable in the creation and implementation of a "data application" web site for health researchers, health educators, and individuals delivering health services.

The applications used in this project were built with standard and advanced hypertext markup language (HTML), combined with the client-side and server-side interpreted scripting commands of "JavaScript" (Gesing and Schneider, 1997; Kent and Kent, 1996; Purcell and Mara, 1997), to present information and collect data of varying types.

Hypertext Markup Language: HTML

Hypertext markup language (HTML) is the "language of the Internet". According to Musciano and Kennedy (1996), and Lemay (1997), HTML is a "document-layout and hyperlink-specification language", which allows web page authors to combine text and graphics in electronic presentations. HTML has a defined code and a specific structure that facilitates the organization of information in an electronic format. Specific electronic HTML readers called "browsers" interpret information that is organized with HTML commands or language code and saved to a computer. (See Figure 1)

Individual's intending to present information to the World Wide Web, prepare the information as a simple text file with HTML commands and then save the document to a designated space (web site address)

on a networked computer. When the Internet-community accesses the designated web site address using the specific "URL: uniform resource locator", the information is interpreted by the "browser" and posted to the user as a "web page".

Interpreted Languages: Scripting

Interpreted computer languages, of which JavaScript is one type, are comprised of "scripts" or commands that are embedded in HTML document files (see Figure 2). Scripts are more complex computer language statements, which invoke computer actions. The rapidly emerging JavaScript language uses a presentation style and structure similar to that of "C" language. However, JavaScript is far less cryptic than "C" or "Java".

In using the interpreted computer languages, scripts are written as part of the published web page. By including the scripts as part of the HTML code pages, the scripts are passed directly to the user (also referred to as the "client"), along with the HTML document when the client accesses the web page during an Internet session. The embedded scripts are implicitly interpreted by the client's browser and processed accordingly. The script commands include data processing routines that allow the web author to add functionality to the client's Internet session (Gesing and Schneider, 1997; Kent and Kent, 1996; Purcell and Mara, 1997). For example, in a non-scripted web page, an author may use simple HTML statements to describe the conversion of body weight from a value recorded in pounds to a value recorded in kilograms using any form of text or tables. However, by using script commands, such as those shown in Figure 3, the web page author can provide a simple calculator that converts the individual's weight entered in pounds to their weight as a value in kilograms. Script commands can range from simple arithmetic operations including logical decisions to complex computations.

JavaScript is among the most popular of the Internet script languages. JavaScript uses a logical, structured approach in presenting command statements to the client's computer. JavaScript has the robustness of more sophisticated computer languages but JavaScript commands are written in simple terms using less cryptic style and jargon.

Compiled Languages and the Common Gateway Interface (CGI)

The corollary to the interpreted language is the "compiled" computer language. Compiled languages are comprised of code that is processed by a compiler residing on the web page author's home server. Compiled languages are used to create specific "stand alone" applications otherwise known as executable programs ("*.exe" or "*.com" programs). (See Figure 4). Neither the compiled code nor the executable program is ever passed to the client. Rather the client passes actions to the web page author's server. These actions include database searching, user registration, or submitting data as in credit card information (Gesing and Schneider, 1997). The actions are passed to the web page author's home server via a pathway referred to as the "Common Gateway Interface" (CGI).

The Common Gateway Interface is a standard that was developed to enable individuals to access specific programs across the Internet. CGI provides interactivity for specific types of HTML documents, especially the input of information on specific data collection forms (Purcell and Mara, 1997; Starr, 1997). Typically, the CGI routes user-entered information to a web page author's home directory where the information is processed. CGI is used extensively in web-based marketing and sales of products over the Internet.

Server-side Scripting Commands

Although CGI commands (e.g. PERL, TCL, C++) provide functionality to the delivery of web pages, CGI commands are cryptic. An alternative to CGI command sequences is server-side scripting. The term server-side refers to the location where the command processing occurs. Simple JavaScript commands embedded into HTML code and processed on the client's computer are referred to as client-side scripts. However, when an individual passes information back to the web page author, such as a credit card number, a database query, or some other form of intentional submission, then they are invoking specific processing commands that run on the web author's computer or server. The scripts or commands that handle the submitted tasks are called "server-side" scripts.

Currently, the availability of software that handles server-side scripting is limited to products released by Netscape Communications Inc. Server-side JavaScript features are bundled into the Server software. Server-

side JavaScript is an application building product that bypasses the traditional CGI. Server-side JavaScript software allows the web page author to develop CGI-like functions with the interpreted commands of JavaScript. Server-side JavaScript based applications allow web page authors to create CGI functionality without actually using CGI commands and compiled programs. The learning curve is considerably reduced when using server-side JavaScript because the server-side scripts are merely extensions of client-side scripts. Therefore, researchers can focus more on **the data collected** than on **the collection of data**. Using server-side scripting, health researchers can request data and provide feedback across the Internet (Purcell and Mara, 1997; Richer and Richer, 1997).

The Model

Figure 5 illustrates the integration of the components of a model for the development of a web-based research application, combining standard and advanced HTML with "client-side" and "server-side" JavaScript to process data over the Internet. The model follows that of a basic feedback loop having three stages: i) input, ii) processing, and iii) output. Each stage of the model contains specific primary, secondary, and tertiary levels.

Input Stage

The items in the input stage combine the features of HTML with the elements of JavaScript. The input stage can be comprised of two independent, primary items: a “request item”, or a “select item”. Either of the two items is required to collect information from the user.

The request item requires the user to provide a specific bit of data for a given question. For example, the “form” feature of HTML was used in Figure 3 to request that the user provide their body weight in pounds. JavaScript variable creation features were then used to convert the input data to the body weight in kilograms, and the result was output onto the HTML form using the variable labeled “outkg”.

The creation of any variable should use two scripting steps. First, the variable is initialized by determining if it exists, and setting the contents of the variable to zero. Second, a label is assigned to the variable. Processing the variable with an arithmetic operation will establish that the contents of the variable are considered as a numeric measurement.

The “select item” is another way to handle input from the user. The select item refers to the selection of an option from a list. The presentation of select items use the “list” features of HTML combined with JavaScript logic statements as illustrated in the code presented in Figure 7. The purpose of the code is to present four options within a multiple choice question. The JavaScript code evaluates the user’s response using simple logic expressions and stores the selection in a variable. The stored value can be returned to the user using HTML statements, or used in a subsequent script in the current web session.

The select item is designed in a multiple-choice format, and the user’s choice is evaluated to produce a score that is held by an assigned variable. For example, consider a user that is presented a question with four options.

1. If the user selects the first option then the contents of the designated variable are assigned a value of 1.
2. If the user selects the second option then the contents of the designated variable are assigned a value of 2.
3. If the user selects the third option then the contents of the designated variable are assigned a value of 3.
4. Finally, if the user selects the fourth option then the contents of the designated variable are assigned a value of 4.

Multiplying the variable contents by the numeral 1 establishes the variable as a numeric measurement. The example given in Figure 5 presents a question about eating behaviours. The user is asked to indicate how often they eat foods of a specific type. Four response choices are presented as "select-items" and a javascript function allows the user to compute their healthy eating score. The basic structure of the JavaScript and HTML codes used to create the "select-item" variables in the webulator presented Figure 6 are explained in Figure 7 below.

Processing Stage

The processing stage of the model is also referred to as the “event handler” or the action stage. The processing stage is comprised of compute items. The compute items handle the information presented by the user in the input stage. The contents of the variables created in the input stage are processed either through logic statements or specific arithmetic operations. As illustrated in Figure 8 the user is presented a web-based calculator which uses request items and select

items. Five input items are used to compute an individual's predicted oxygen consumption potential (ml/kg•min^{-1}): i.) body weight, ii.) age, iii.) time to walk one mile, iv.) heart rate at the end of a timed one mile walk, and v.) gender.

The user's body weight, which is requested in pounds, is converted to a kilogram value by an arithmetic operation written in JavaScript. The variable age is entered as an integer, while time to walk one mile is entered as a decimal (ratio score). Post-exercise heart rate is a ratio score because it is entered as beats per minute. However, indicating whether you are male or female is a nominal based "select item" that is evaluated by logic statements written in JavaScript.

The compute item uses arithmetic operations in the processing stage to handle the values held by the request items. The choice of compute statements is determined by the client's choice of select items. That is, in this example, the compute statements will differ depending on the client's indication that they are male or female. The result of the events in the processing stage produce the user's "predicted oxygen consumption potential" (VO_2 max score in ml/kg•min^{-1}). The entire computational procedure is referred to as an event, and the series of statements to produce the VO_2 max score is referred to as the event handler.

Output Stage

The output stage of the design model is comprised of a variety of feedback items based on the results from the input and processing stages. The feedback items can take several forms. For example, i.) Secondary windows can be generated from main web pages to post new HTML code; ii.) In-line script statements can be added to advanced HTML frames statements to post a graphical representation of data that was collected by the input stage, and computed in the processing stage; and iii.) Uniform resource locators (URL) can be posted to web pages following the processing of direct user input to link the client to web-sites (pages) that contain information pertaining to a particular response.

However, the most important feature of Internet data processing for health researchers is that a web page author can use JavaScript commands to capture data. Common gateway interface routines which use server-side JavaScripts (Richer and Richer, 1997), or PERL scripts (Purcell and Mara, 1997) can be used to write client supplied input data, or the products of the compute items, to the web page author's home server. The ability to use a scripting language to write output to the web page author's home server, based on client-side input data is a formidable accomplishment and a tremendous benefit for researchers.

Combining JavaScript, HTML and CGI processing applications, the researcher can reduce the costs of data collection by posting questions or requesting specific measures on the client's web site. The client responds by selecting items or entering information, which is in turn processed by compute items. The task of data collection is handled by standard JavaScript commands, by writing specific products of the compute items to the web page author's home server. The server-side data-capture-data-write commands are illustrated in Figure 8. The sequence of commands that facilitate the write process allows the user to build applications which combine the three stages of the model (input - processing - output).

Post-web applications are readily applied to the output data set. Data collected across the web is written in standard ASCII format that enables the web page author to process the data through any number of end-user data processing packages (e.g. statistics packages like SAS or SPSS; or data base routines of ORACLE or Informix).

The Application

A group of Year 2 undergraduate students (N=106) enrolled in an Exercise Physiology class were required to complete a standardized walking test as a part of one of the regularly scheduled weekly laboratory assignments. The students were required to walk as quickly as possible around a one mile measured course. The students recorded their weight immediately before the test, the time required to complete the one-mile walk, and their immediate post walk heart rate. The students next logged onto the Internet address where they were directed to a web based calculator, "the AWebulator" shown in Figure 7.

The students entered their data into the Webulator to determine their "predicted oxygen consumption potential" (VO_2 max score in ml/kg_min^{-1}). The result of the web-based calculation was reported directly to the client. However, in addition to direct client feedback, the Webulator included server-side JavaScript commands that presented the option to submit the data to the Instructor's database (Figure 9).

Conclusion

A summary of the events to collect, process, and feedback data, by combining HTML, JavaScript, and a third party software that crosses the CGI are mapped in Figure 10 and Figure 11. An Internet-based data processing system, such as that described in this paper will have positive implications across a variety of research areas. Costs associated with production and distribution of data collection instruments will be reduced since printing costs are virtually non-existent. Time, expenses, and errors, associated with data entry will be reduced by having fewer individuals handle the data. Speed of analysis will increase because the data will reside in a computer readable format following submission of the form data by the client. Finally, direct feedback to the client can be provided during the data collection stage, which will essentially minimize the need for post-project debriefing sessions.

A sample of the output file is presented in Figure 8. Notice that in addition to the client's surname, student number and gender, the output data set contains the results for two different VO_2 maximum computational formulae presented as feedback items in Figure 7 (Kline, Porcari, Hintermeister, Freedson, et al, 1987; Dolgener, Hensley, Marsh, and Fjelstul, 1994).

References

Collis, B., (1996). The Internet as an Educational Innovation: Lessons from Experience with Computer Implementation. *Educational Technology*, November-December, 21-30.

Dolgener, F., Hensley, L., Marsh, J., Fjelstul, J., (1994). Validation of the Rockport Fitness Walking Test in College Males and Females. *Research Quarterly for Exercise and Sport*, 65(2), 152-158.

Gesing, T., Schneider, J., (1997). *Javascript for the World Wide Web*, Berkeley: Peachpit Press.

Kent, P., Kent, J., (1996). *Official Netscape Javascript Book*, Research Triangle Park: Ventana Communications Group, Inc.

Kline, G.M., Porcari, J.P., Hintermeister, R., Freedson, P.S., Ward, A., McCarron, R.F., Ross, J., Rippe, J.M., (1987). Estimation of VO2 max from a One-mile track walk, gender, age, and body weight. *Medicine and Science in Sports and Exercise*, 19, 253-259.

Maddux, C.D., LaMont-Johnson, D., (1997). The World Wide Web: History, Cultural Context, and a Manual for Developers of Educational Information-based Web sites. *Educational Technology*, September-October, 5-12.

Musciano, C., Kennedy, B., (1996). *HTML, The Definitive Guide*, Bonn: O'Reilly & Associates, Inc.

Purcell, L., Mara, M.J., (1997). *The ABCs of Javascript*, San Francisco: Sybex Inc.

Richer M., Richer, J., (1997).*Official Netscape Livewire Book*, Research Triangle Park: Ventana Communications Group, Inc.

Starr, R., (1997). Delivering Instruction on the World Wide Web: Overview and Basic Design Principles. *Educational Technology*, May-June, 7-15.

Stemler, L.K., (1997). Educational Characteristics of Multimedia: A Literature Review. *Journal of Educational Multimedia and Hypermedia*, 6(3/4), 339-359.