*Behavior Research Methods, Instruments, & Computers*
*1998, 30 (2), 306-308*

## EXPERIMENTS ON THE WEB

# CGI scripts: A strategy for between-subjects experimental group assignment on the World-Wide Web

ROBERT H. MORROW and ADAM J. McKEE
*University of Southern Mississippi, Hattiesburg, Mississippi*

Psychological studies implementing the World-Wide Web as a data collection medium have traditionally been constrained to data obtained from surveys or from within-subjects comparisons. Common gateway interface (CGI) scripts, however, provide researchers with a means of collecting data for between-subjects comparisons. This paper provides a description of the information flow between the user (subject) and the experimenter by referencing a completed between-subjects assignment CGI. A current online cognitive psychology experiment serves as an example of the utility of CGIs as subject assignment mechanisms. Server-side and client-side scripts are compared for effectiveness at data collection, with these types of designs in mind.

The ubiquity of the World-Wide Web (WWW) has served to increase the rate of data collection for many psychologists. A testament to this observation is the increase in both degree and kind of data collection devices currently found on the Internet. Whereas survey forms (Schmidt, 1997) and within-subjects research methods implementing static hypertext markup language (HTML) pages have become a quotidian staple in the experimental psychologist's research toolbox, methods involving between-subjects designs have been less recognizable as Internet-based experimental designs.

Central to the implementation of a between-subjects experimental design on the Internet is a program that is external both to the user/subject and to the experimenter. A method is necessary that *instructs* the subject about the experimental requirements, as well as provides accurate, consistent data reporting to the experimenter. Server-side scripts that are written as common gateway interfaces (CGIs) can serve this function.

## Server-Side and Client-Side Scripts

The key differences between a server-side and a client-side script are not so much a matter of what is being done as of where it is being done. Although we may think of each as containing the instructions for an experimental protocol, there exist key differences between the two. Client-side scripts occur on the machine receiving feedback from the server. As a matter of function, client-side scripts are *shipped* in their entirety to the computer (more specifically, the browser) that is downloading the experiment. The client's browser then builds the experiment each time it is requested from the server. Results of the experiment are then shipped back to the server. The scripted actions of the client-side script take place entirely on the subject's/client's machine. As a methodology for experiments, this saves bandwidth and reduces confounds that are due to temporal insensitivity (i.e., geographical proximity to the experimenter's Web server). Server-side scripts, on the other hand, perform all actions on some web server remote from the user, after interacting with the user's computer. As such, the scripts are independent of what each individual computer and the user do at the other end. To be sure, the user's actions influence the processing that takes place on the server through server-side scripts, but usually more indirectly than client-side scripts.

Experiments involving highly sensitive measurements, such as reaction time or latency, are not precluded from the Internet as a medium for psychology experiments, but such experiments are best implemented as client-side scripts. As an example of a highly sensitive client-side reaction time experiment for which the source code is readily available, see Link's (1998) Stroop demonstration. Although the processing speed of any single user's computer could influence the data collected when a client script is used, the script is independent of the amount of traffic on the Internet at the moment of data collection, since the program has been executed only after being completely built by the client browser. Inasmuch as the client computer is the target for the experiment, the experimenter must also be cognizant of the browser that the subject uses on the client computer. This consideration warrants the use of a client-side scripting language that accommodates most (if not all) client computers. At the time of this writing, JavaScript is the only language that accomplishes this

Correspondence concerning this article should be addressed to R. H. Morrow, U.S.M.P.O. Box 5934, Hattiesburg, MS 39406 (e-mail: rmorrow@ocean.otr.usm.edu).

task in both a UNIX and a Microsoft windows server environment. Although other client-side languages (such as Microsoft's VBscript, ActiveX technologies) have been found to produce idiosyncratic processes specific to a popular browser, Java and JavaScript correctly process client-side data from Netscape (Version 3.0 or later), Internet Explorer (Version 3.0 or later), and Mosaic.

Server-side scripts, by way of contrast, can be thought of as a moderator between the client computer (and, hence, the subject) and the experimenter. The key advantage to server-side scripting is that the data and processes of server-side scripts are external to the client browser and to the static HTML page that the experimenter utilizes to provide feedback to any single subject. Scripted processes that run on the server can be directed to provide user feedback, implement a selection structure (for example, assignment to an experimental group), send data to a file or an e-mail account, or implement server-specific methods (such as a date or time stamps). As a basis for consistency of experimental conditions, then, server-side scripts provide a foundation for the between-subjects experiment. Some sample server-side scripting languages include Microsoft's Active Server Pages (for Microsoft servers, such as O'Reilly's WebSite or Microsoft's Personal Web Server), C/C++ (runs on multiple platforms), or Perl. Perl is the language chosen for this demonstration, since that language ports easily to any UNIX system. UNIX systems are currently the most ubiquitous web servers in existence, so this demonstration is specific to that web server environment.

Although the term *Common Gateway Interface* is not synonymous with server-side scripting, for UNIX-based system accountholders it might as well be. The CGI is a protocol written explicitly for the UNIX operating system that allows users to submit data from HTML pages to an e-mail account or to append that data to a file using the GET or POST methods (Kieley, 1996). GET and POST Perl CGI methods differ in that the GET method appends small amounts of data (such as the subject's name, age, and demographic descriptors) to the URL that the browser sends back to the web server (Lemay & Danesh, 1997), whereas the POST method sends the data separately from the call to the CGI script and usually places that data in a temporary file until the server is ready to process the data by utilizing some selection structure. For a simple but thorough description of this process, see Herrman's (1996) "Teach Yourself CGI Programming With Perl 5 in a Week."

## Sample Experiment Using a Perl-Written Common Gateway Interface

A cognitive psychology experiment (Morrow, 1998) that instituted a CGI written in the Perl language that examines the context effects of the popular, ambiguously written Bransford and Johnson (1973) reading "Washing Clothes" shall be detailed in the following paragraphs. The experiment was an attempt to ascertain whether there is an interaction between a nontitled reading of "Washing Clothes" and memory decay (measured with a word-fragment cued recall test over various time intervals covering 3 days). This group was to be compared with a full-text reading administered in the same fashion.

In that experiment, the subjects log on to a standard HTML page at their own convenience and provide identifier data, demographic data, and the instructor's name (in order to assign extra class credit for experimental participation). The subject then submits this data. The data submission *calls* the CGI script to process an additional subject. The CGI examines the contents of a separate counter file maintained on the web server. On the basis of the number of subjects that have previously logged on, the CGI assigns that subject to one of two paragraph readings (with or without the "Washing Clothes" title) and increases the number referenced in the counter file by one. After completing the reading, the subject selects a *next* link from the HTML page. This link recalls the subject's assignment number and requests of the subject that he or she return at some time that has been determined on the basis of that number, in order to complete a word-fragment measure. Some possible subject return times include immediately, 2–4 h later, 4–8 h later, 8–16 h later, 1 day later, 2 days later, and 3 days later. See Figure 1 for an outline of events in the experiment's protocol, from the subject's initial logon to experiment completion.

The described CGI is available at http://ocean.otr.usm.edu/~rmorrow/ComGatIn.html and was designed specifically for this experiment. It may be used, however, for any between-subjects experimental design on the WWW. Although the logic used in the script's selection structure is rudimentary, in that the primary logic is based on the *if-then-else* selection structure, this is intentional, so that novice programmers may adapt the script to suit their individual needs. More experienced programmers may prefer implementing an array (for a sample alternative script that uses such an array, provided by reviewer Alan Schwartz, visit http://ocean.otr.usm.edu/~rmorrow/AltComGat.html). All of the experiment's output may be sent to an e-mail account of the experimenter's choosing by substituting the mail address in the script for one directed to that person's e-mail address. Of course, all of the data may be sent to a data file on the server, as well as or instead of an e-mail reporting of the data.

## Useful Tailoring of the CGI Script for Other Experimental Designs

The script that operates the selection/assignment structure for the above-described experiment may be modified so as to accommodate other between-subjects experimental designs. For example, the selection structure is designed to provide linear assignment to the group conditions. This could be modified by selecting a random number generator, using the Perl scripting language. Some applicable random number generators may be found in the Oasis Archives' site "Truly Random-Perl Interface to a Truly Random Number Generator Function Archives" (1998). Another possible experimental protocol that may be modified is the number of groups and type of feedback provided to the user as a result of that selection. The script has been written in *if-elsif* logic in order to accommodate such alterations.

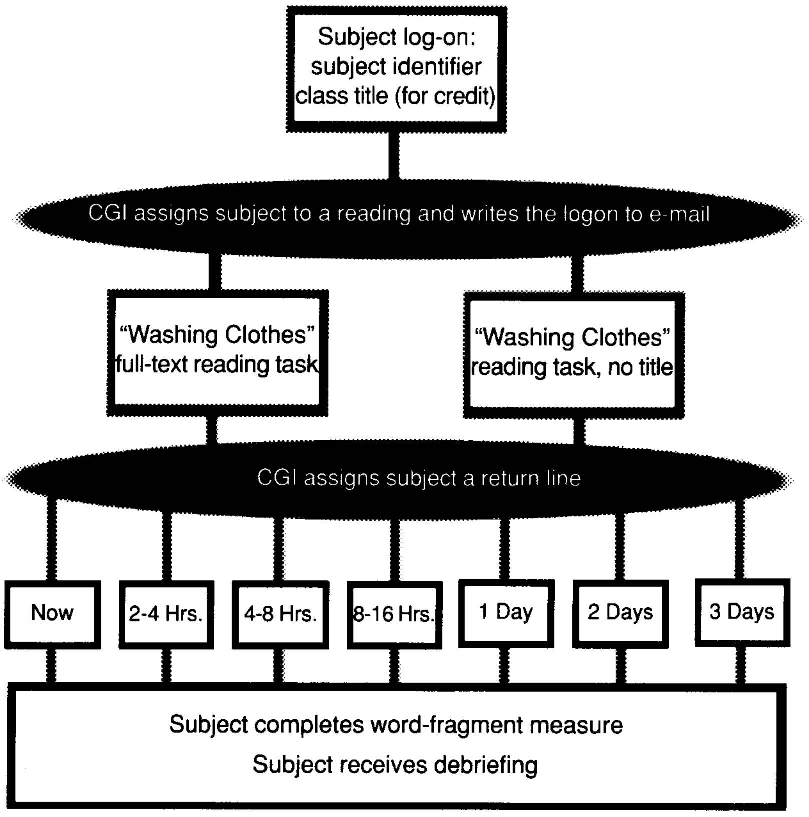
Subject log-on: subject identifier class title (for credit)
CGI assigns subject to a reading and writes the logon to e-mail
"Washing Clothes" full-text reading task
"Washing Clothes" reading task, no title
CGI assigns subject a return line
Now
2-4 Hrs.
4-8 Hrs.
8-16 Hrs.
1 Day
2 Days
3 Days
Subject completes word-fragment measure
Subject receives debriefing

**Figure 1. Event flow from subject log-on to experiment completion.**

By simply extending or abbreviating the number of elsif statements in the selection structure (perhaps through cut-and-paste functions), the experimenter provides an adequate number of experimental conditions to account for his or her design. The feedback provided to the subject may be altered in a similar manner. In short, the provided CGI is a functional template for between-groups designs.

## Conclusions

Although the Internet has been a breeding ground for the development of within-subjects experimental designs, between-subjects designs have been less well represented. This underrepresentation seems to be largely a function of the lack of availability of CGI scripts. Since server-side scripts are a central element in the design of within-subjects psychological experiments, a UNIX-friendly script has been explained in terms of user interaction with the elements of server-side processing. The event flow from subject logon to completion of a data collection episode has been explained in order to clarify the processing actions that can occur in this design.

The CGI used in the described experiment was written in novice-level selection structure, where possible, in order to facilitate CGI use by experimenters who may not be as familiar with Perl or with server-side scripting. Although the experimenter should have programming experience in some language, a complete familiarity with the Perl language is unnecessary if the experimenter alters the referenced scripts to fit the design of the experiment. In addition, the script has been explained to be useful in a UNIX/Linux environment; developers with other server types (Microsoft-specific servers) will want to pursue another scripting language. UNIX/Linux servers are, for the moment, the most widely implemented servers in the academic world. Perl, then, should be compatible with most psychological between-subjects experimental designs.

## REFERENCES

Bransford, J. D., & Johnson, M. K. (1973). Considerations of some problems of comprehension. In W. G. Chase (Ed.), *Visual information processing* (pp. 383-438). San Diego: Academic Press.

Herrman, E. (1996). *Teach yourself CGI programming with Perl 5 in a week.* Indianapolis: Sams.Net

Kieley, J. M. (1996). CGI scripts: Gateways to World-Wide Web power. *Behavior Research Methods, Instruments & Computers*, **28**, 165-169.

Lemay, L., & Danesh, A. (1997). *Teach yourself Web publishing with HTML 4 in 14 days, second professional reference edition.* Indianapolis: Sams.net

Link, R. (1998). *Stroop demo* [On-line]. Available: http://www-psych.nmsu.edu/~rlink/stroop1/

Morrow, R. M. (1998). *Online psychology experiment* [On-line]. Available: http://ocean.otr.usm.edu/~rmorrow/experiment.html

Oasis Archives (1998). *Truly random-Perl interface to a truly random number generator function archive listing* [On-line]. Available: http://www.oasis.leo.org/perl/exts/math/Math-TrulyRandom.dsc-list.html

Schmidt, W. C. (1997). World-Wide Web survey research: Benefits, potential problems, and solutions. *Behavior Research Methods, Instruments, & Computers*, **29**, 274-279.

(Manuscript received December 10, 1997;
revision accepted for publication March 6, 1998.)