Chapter 20: Advanced Techniques for Experimentation on the Web

This chapter reviews advanced techniques that provide additional programming control over tests and experiments on the Web. As you have seen in Chapters 5-16, virtually any study that can be done with paper and pencil materials, including those with pictures and graphics, can be done with HTML forms. In Chapter 14, you learned some additional techniques for working with graphics and sounds.

Advanced methods for Web research, presented by researchers who have "pioneered" these techniques, are presented in the book edited by Birnbaum (in press). The authors in that book present much good advice that has been learned from experience. These chapters also include advanced discussions of JavaScript, Java, Authorware, and Server-Side programming to achieve greater control over Web experiments. This chapter will review these techniques briefly, so that you can determine if advanced study of these topics is for you.

As you learned in Chapters 17-19, JavaScript allows one to create experiments that interact with the participant, so that the experiment can tailor itself to the unique pattern of responses a participant might give. JavaScript can be used to time events, to score or tabulate responses, and to provide feedback to the participant. Because JavaScript can be included in a Web page, it can make the process of research more open and public. One experimenter can learn from and build on another's work.

A. Advanced JavaScript Techniques

There are times when one does not want information to be too openly available. By the use of frames, it is possible to make the JavaScript code nearly "hidden" (difficult for the typical participant to find). This technique and other useful tricks for using

JavaScript to randomize and control Web questionnaires and surveys are presented in the chapter by Baron and Siepmann (in press). Examples associated with their chapter can be found at URL (http://www.psych.upenn.edu/~baron/examples/), which is linked from the list of examples. In experiments, there is no great motive by participants to "hack" the JavaScript, so these techniques are quite adequate for most psychological research where one does not want the participant to be able to easily view the code of the study.

However, there are situations in which one does not want information to be available at all. For example, if one was giving an exam in a class and the key was contained in the page, there would be a motive to open the page and decode the key. In addition, there are experiments in which graphics must be created in response to the user's input. These situations can be handled by server-side programming. Graphics can be created and controlled by Java Applets.

A combination of JavaScript and Java is used by the on-line museum of perception and cognition (Lange, 1999). This museum can be accessed at URL [http://www.ulb.ac.be/psycho/museum.html]. An abbreviated section of the museum is available in English; to reach it, click on the English flag. Housed in Belgium, documents in this Website are available in French, but even those available only in French are well worth your patient efforts to understand.

The article by Lange (1999) discusses useful techniques of JavaScript. A collection of JavaScrips associated with her article (with explanations in English) can be downloaded from the following URL [http://www.ulb.ac.be/psycho/brmic.html].

B. Java Programming

Java is an advanced programming language that should not be confused with JavaScript, despite some similarities. Java can be used to write stand-alone programs that will run on any computer with the appropriate interpreter (software). Java can also be used to write applets that can be sent along with Web pages in the much the same manner as images are included in a Web page. Whereas JavaScript programs can be sent as source code and compiled on the client's (reader's) computer, Java applets are precompiled by the developer (programmer) and supplied as separate files of symbolic codes (bytecodes) that are sent like files of images or sounds along with Web pages. Like JavaScript, Java programs are intended to run on any type of machine running the appropriate browser.

Advantages of the Java language for cognitive psychology experiments are described by Stevenson, Francis, and Kim (1999) and by Francis, Neath, & Surprenant (in press). Their Cognitive Psychology Laboratory at Purdue University, which makes use of Java (and JavaScript) can be reached at the following URL [http://www.psych.purdue.edu/~coglab/index.html].

To use the OnLine Laboratory, one must have both JavaScript and Java installed and enabled in a modern browser. A test page is provided at URL [http://www.psych.purdue.edu/~coglab/testpage.html]. This test page will let you know if your browser is properly configured to use the lab. Two of the experiments discussed in Francis, et al (in press) include the Brown-Peterson Memory Task, at URL http://www.psych.purdue.edu/~coglab/BrownPeterson/BP.html] and the Sperling Partial Report task at URL [http://www.psych.purdue.edu/~coglab/PartialReport/PR.html].

By early 1999, there were 15 on-line projects with studies of perception (e.g., apparent motion, visual search), memory (e.g., false memories, serial position), neurocognition (e.g., receptive fields, brain asymmetry), and language (e.g., lexical decision). Each project is also introduced with background material to help the student understand each research area.

A Web page of resources for learning more about Java, including a sample reaction time program, is located at the following URL:

[http://www.psych.purdue.edu/~coglab/java.html].

Ch20_ex1.htm illustrates how to include a Java Applet in a Web page:

```
<HTML><HEAD>
<TITLE> Test of Java</TITLE></HEAD>
<BODY>
<H3>TEST OF JAVA</H3>
IF JAVA WORKS, THEN YOU WILL SEE "HI" BELOW :<BR>
<APPLET CODE="Ch20_ex1.class" WIDTH=200 HEIGHT=50 ALIGN=top>
Sorry, Java is not installed.
</APPLET>
</BODY>
</HTML>
```

The new tags in this example are `<APPLET></APPLET>`. Between these tags can be written text that will display if Java is *not* available. `CODE="`*`program`*`.class"` identifies the file containing the compiled Java applet, or *code*. In this case, the file containing the program's byte codes is *Ch20_ex1.class*. The other attributes (`WIDTH, HEIGHT, ALIGN`) are just like those in the IMG tag. In this example, the applet is 200 by 50 pixels.

Programs in Java are written on a text editor, and saved with the extension `.java`. The source code applet in *Ch20_ex1.java* is as follows:

```
import java.awt.Graphics;

public class Ch20_ex1 extends java.applet.Applet{

    public void paint(Graphics g){
         g.drawString("Hi! Java is Working!", 5, 25);
   }
}
```

Like JavaScript, Java is case sensitive. The first line imports graphics. The second line names the applet *Ch20_ex1*, makes it public, and makes it an applet, extending the more general category of applets. The line that does the work is `g.drawString()`, which as you might guess, writes the string in quotes to the screen.

Each statement in Java ends with a semicolon, even if there is only one statement on a line.

This Java program is saved as a text file with the name *Ch20_ex1.java*. The filename must match the program name, and it must have the `.java` extension. This file is included in the examples on CD and can be examined in a text editor. The Java program must then be compiled by the *Java Compiler*, which can be downloaded for free. (A link to the download site is included on the CD.) The compiler converts the program to byte codes, which in this case, were saved by the compiler in a file named *Ch20_ex1.class*. The Web server delivers the HTML and this file of byte codes.

In summary, this example uses three files: *Ch20_ex1.htm* is the HTML that calls the Java Applet, *Ch20_ex1.java* is the source program (a text file), and *Ch20_ex1.class* is the file of byte codes. To make the program available on the Web, only the HTML and the file of byte code files (*Ch20_ex1.htm* and *Ch20_ex1.class*) need to be placed on the server.

The next example illustrates some similarities (and differences) between Java and JavaScript. *Ch20_ex2.htm* contains an applet like the JavaScript example in Chapter 18 Example 6, which calculates interest by means of a loop. The source code, *Ch20_ex2.java*, is as follows:

```
import java.awt.*;

public class Ch20_ex2 extends java.applet.Applet{

    public void paint(Graphics g) {
      float debt = 10000.0F;  // declares debt to be floating point
      float rate = .08F;          // declares rate to be floating point
      for (int i=1; i <= 15L ; i=i+1)   {
      debt = debt + rate*debt;
      g.drawString("\n debt = " + debt, 10, i*25);
                                          }
                                    }
                                                 }
```

In Java, variables must be declared (typed). Expressions such as `float debt` and `int i` declare `debt` and `i` to be floating point number and integer variables, respectively. The `for` loop is similar to that seen in JavaScript.

Java is an object oriented programming (OOP) language that supports inheritance. The technique in Java (and OOP languages in general) is to create very general programs from which special cases can inherit their main characteristics and also have unique features special to the particular situation. An examination of the experiments at the Cognitive Psychology Laboratory at Purdue will convince you of the power of this approach.

C. Use of Authorware/Shockwave Techniques

Instead of using a programming language, it is possible to construct cognitive psychology experiments that can control stimulus timing and measure response times by means of higher level software that relieves the designer of most programming. This approach is described by McGraw, Tew, and Williams (in press), who use Authorware by

Macromedia to implement a number of classic cognitive psychology experiments (see also Williams, McGraw, & Tew, 1999). The experiments require the Authorware Player (Shockwave) plug-in, which is free from Macromedia. These experiments can be found at URL [http://www.olemiss.edu/PsychExps/], along with information about how to download and install the free player. It is possible to select the "sampler," which allows you to experience briefly each of the experimental projects available on line. This cooperative, shared research site also allows instructors to use the lab to collect and download data. Information for instructors who wish to join the lab is available in the Web site.

An advantage of the Authorware approach is that it requires much less programming ability from the users. The program has a graphic interface that allows experiments to be constructed by moving and inserting tasks and displays on a flow line that characterizes the experiment. A disadvantage of the approach is the high price of the Authorware software. Although the player is free, the authoring software costs over $600 in 1999.

Another Web site that makes use of a plug-in is the Internet Psychology Lab (IPL) at the University of Illinois, Urbana-Champaign, which can be accessed via URL [http://kahuna.psych.uiuc.edu/ipl/]. This lab uses Java 1.1, JavaScript, and the IPL plug-in, which is available for Windows 95 and NT. The lab offers demonstrations of Visual Perception (including such paradigms as Signal detection and selective adaptation and illustrations of illusions such as the Mueller-Lyer, Ponzo, Horizontal-Vertical, and others). In Auditory Perception there are demonstrations of pitch perception, the Shepard

tone, and others; in Cognition, there are demonstrations of basic reaction time, choice reaction time, Stroop effect, Chimeric faces, and others.

D. Server-Side Programming to Control Experiments

JavaScript, Java, and programs utilizing plug-ins have in common that the program is delivered by the server (in one form or another) to the reader's (client's) browser, and the computing is carried out on the reader's (client's) computer. This approach has the advantage that it minimizes the load that is placed on the server. This approach has the disadvantage that the reader must have the proper configuration of computer, browser, and plug-ins to make the software work. Ideally, such software would run equivalently on all machines and all browsers, but in practice, different people will have different experiences. For some people your experiment might run slowly or erratically, and for others, it will not run at all. Furthermore, by sending everything to run on the client's computer, a measure of control and security is lost.

Another approach is to run some or all of the experiment on the server itself (Morrow & McKee, 1998; Schmidt, 1997a, in press). That guarantees that the program will run, it secures certain aspects of the experiment that must remain secure, and it means that the computer and software of the end user will have little effect on the process. It has the disadvantage of placing a burden on the server to make computations as well as serving files. That can slow down transactions with the server. When delays are significant, some participants will lose patience and quit the experiment, thinking that your site has frozen up. If the server is fast, the programs efficient, and the traffic light, these delays should not be a problem.

There are many tasks that can be done either by server-side programming or by programs that run on the reader's (client's) computer, such as JavaScript programs. For example, Alan Schwartz independently developed a program to carry out Bayesian calculations like those of the JavaScript Bayesian calculator in Chapter 16, but his program was written in Perl and runs on his server. His calculator can be found at URL [http://araw.mede.uic.edu/cgi-bin/testcalc.pl]. For his tutorial on Perl, see Schwartz (1998). You can compare the speed of the server side program with the JavaScript calculator in Chapter 16 to get an idea of this aspect of server side programs.

Some tasks, however, are best performed or only possible on the server (Schmidt, in press). For example, programs that save data to the server and programs providing password protection must be done on the server. For obvious reasons, you wouldn't want to allow remote users to be able to save to your disk or control your passwords!

In the projects described in this book, the CGI programs that organize and save the data were created by PolyForm (see Appendix A). The server can also be programmed to check data consistency, check for response omissions, check for multiple submissions from the same person, score tests, give feedback to participants, and many other useful tasks (Schmidt, 1997). Schmidt (in press) describes how *server side includes* (SSI), programs that run on the server and augment the experiment, can be used to provide additional functionality or control to an experiment.

Suggestions for setting up and running your own server are given in Francis, et al. (in press), and Schmidt, Hoffman, and MacDonald (1997). Francis, et al (in press) describe how an abandoned 486 machine can be brought back to life as a useful Web server, using server software that is freely available on the Web. This approach is very

inexpensive, but requires expertise (and effort) on the part of the reviver. Commercial packages including both hardware and software for a Web server are available (in 1999) for about $3500.

E. Summary

This chapter reviewed advanced techniques that can be used to add considerable power and control to a Web experiment. JavaScript is a powerful scripting language that can be used to create interactive experiments in which time can be controlled or measured. Java is a powerful language that can be used to create stand-alone applications or applets that can be sent with Web pages. This language can be used to control and dynamically change graphics. Basic techniques for including a Java application in a page were described. The Authorware approach, in which one can use a high-level program to develop experiments, requires less programming expertise to create experiments similar to those that can be implemented via Java. In addition, the role of server-side programming in Web experiments was described. Server-side programming can accomplish the same tasks as Java and JavaScript, and it can also do tasks, such as saving data, that only the server can do.

F. Exercises

1. Read Lange (1999) and download the example JavaScripts that accompany that article. Study those examples.
2. Read Baron and Siepmann (in press) and explore the JavaScript examples in their Web site. Create a short questionnaire that implements their techniques.
3. Read Francis, et al (in press), visit the Cognitive Psychology Lab at Purdue, and participate in two experiments of your choosing.

4. Download the Java Development Kit from Sun. Compile applets from the source code for Ch20_ex1.java and Ch20_ex2.java. Test them in your browser.
5. Read McGraw, et al. (in press) and visit PsychExps at Ole Miss. Participate in the sampler to see how the experiments are implemented at that site.
6. Project idea: Examine the Java Resources page at Purdue. Develop a Reaction Time program that extends the sample program available at that site. Change the program so that the stimulus will appear (randomly) in either the left or right side of the visual field, and the participant must push "F" if the stimulus appears on the left, and "J" if the stimulus appears on the right. Allow the experimenter to control the probability that the stimulus appears on the left or right.
7. Project idea: Acquire Authorware from Macromedia. Develop the same experiment described in #6 by means of Authorware.
8. Project idea: Read articles on running your own Web server (Schmidt, et al, 1997; Francis, et al, in press). Bring an old 486 machine back to life as a Web server, using the methods described by Francis, et al (in press).
9. Download Perl for free. Search the Web for Perl resources. Learn Perl and write SSIs to do one of the following tasks: random assignment of participants to conditions, check for multiple submissions from the same remote address, check for missing data, check for submissions from a referring page other than your intended Web page.