Behavior Research Methods, Instruments, & Computers
1999, 31 (1), 34-45

PERCEPTION, CONDITIONING, AND ACTION

Museum of perception and cognition website: Using JavaScript to increase interactivity in Web-based presentations

MARIELLE LANGE
University of Brussels, Brussels, Belgium

The present paper introduces the Museum of Perception and Cognition website. This site offers an interactive introduction to cognitive psychology via a JavaScript-based illustration of optical illusions and a Java-based presentation of experimental paradigms. Its content and utilization as lecture support for 1st-year students at Free University of Brussels is described. This paper also endeavors to share experience we gained in Web-based lecture materials development. It introduces the Web lecturer with JavaScript features and utilization and provides him/her with a description of reusable JavaScript routines downloadable from our site that relate to more engaging, interactive, and effective Web-based presentation of course materials.

In 1997 the Laboratory of Experimental Psychology at the University of Brussels initiated the Museum of Perception and Cognition project, based on content presented via the World-Wide Web. This site offers an interactive introduction to cognitive psychology using a JavaScript-based illustration of optical illusions and a Java-based presentation of experimental paradigms. The Web format was primarily chosen for accessibility, ease of use, flexibility, and platform independence. This format also enabled us to explore how Web-based presentations and their specific capabilities could be used for developing course materials: (1) combining graphics, tables, forms, video, and sound to structure the information in a variety of ways; (2) including links that take students to resources from around the Web; and (3) implementing interactivity with the aid of JavaScripts and Java applets.

In this paper, I first discuss the issues of interactivity and user control in computer-based presentations. Then, the site content and its utilization as lecture support for 1st-year undergraduate psychology students at Free University of Brussels are described. This description illustrates how the Web medium can, with the help of JavaScript and Java capabilities, be turned from a poor text browser into a stimulating interactive interface. In the second part of the paper, I share experience gained in JavaScripting and introduce the candidate Web lecturer to JavaScript features and utilization. This description is supplemented by examples of JavaScript routines that relate to more engaging, interactive, and effective Web-based presentation of course materials. The purpose of this description and development of library routines is to equip the psychologist with better tools for Web-based presentations—tools that are, because JavaScript is easy to learn and use, accessible to most psychologists.

INTERACTIVITY AND USER CONTROL IN HYPERMEDIA PRESENTATIONS

The term *multimedia* refers to the integration of several different media into one delivery format. For instance, audio, video, text, graphics, and sound can be put together on CD-ROM or on the Web. However, in the Web environment, where these media components are linked with hypertext, the term *hypermedia* is applied. When hypermedia has the ability to respond in variable ways to user actions, it is called *interactive hypermedia* (O'Reilly & Morgan, 1997).

On account of these considerations, interactivity is not just a descriptor for any multimedia product; even if it is clear that a Web-based presentation adds richness of content, it is not by itself a guarantee of better teaching (or student learning). Kennedy and McNaught (1996) stated that "the click-and-point model that is at the center of much of course-related hypermedia presentations currently produced encourages surface learning rather than deep learning, and is likely to favor rote learning rather than understanding." Furthermore, Peterson and Orde (1995)

M.L. is an F.N.R.S. mandatory of the National Fund for Scientific Research (Belgium). She gratefully acknowledges the help of the University of Brussels Network Service (Resulb) in setting up this project and the contribution of many members of the Experimental Psychology Laboratory to the content of the site; more specifically, she gratefully acknowledges the contributions of Monique Radeau, José Morais, Daniel Holender, and Paul Bertelson. She also thanks Alain Content for useful comments on previous versions of this document. Correspondence should be addressed to M. Lange, Laboratoire de Psychologie Expérimentale, Université Libre de Bruxelles–CP 191, Avenue F. D. Roosevelt, 50, B-1050 Brussels, Belgium (e-mail: mlange@ulb.ac.be).

reported that students often did not complete multimedia programs that used the computer as a book with too much text and no capacity for creative interaction. In contrast, it has been established that increasing learner control can improve student motivation and interest in content (Laurillard, 1993; Ramsden, 1992). But this is more than allowing the user to simply choose between a set of options or to turn pages of cute animations. Sims and Hedberg (1995) identified several critical dimensions: control over content (i.e., selection of topics), control over pacing (i.e., the speed and time at which content is presented), control over sequencing (i.e., the order in which the content is viewed and the extent to which it encourages exploration), method of presentation (i.e., text- or graphic-based explanation), and provision of optional content (i.e., access to additional information upon request). Two other features that are known to extend student engagement in the learning process are feedback about learning (i.e., evaluation and auto-assessment; Ramsden, 1992) and attractiveness of the presentation (i.e., it should be visually stimulating; Gazzard & Dalziel, 1997).

SITE CONTENT

The museum home page can be reached via the URL [http://www.ulb.ac.be/psycho/museum.html].[1] As is apparent from its name, the museum website addresses the topics of perception and cognition. Each topic is allotted a different section. In the section about perceptual processes, the theory behind optical phenomena is introduced starting from a set of JavaScript-based dynamic presentations of visual illusions. In a second section, cognitive processes are illustrated with Java-based interactive demonstrations of some classical experimental situations. Materials presented in both sections are targeted to 1st-year students as support for their experimental psychology lecture. The section on perceptual processes was designed primarily for use as a source of illustration in a classroom situation, whereas the cognitive processes section was designed as a stand-alone resource that students could access in their leisure time. Though the site was conceived with 1st-year students in mind, it was meant to be of interest to other students as well. Accordingly, the purpose of these Web-based materials was to reinforce, rather than replace, lecture courses and provide students with useful resources such as on-line papers, tutorial sites, bibliographical databases, and so on.

Perceptual Illusions Section

Each illusion and each theoretical chapter is presented on a separate page, the whole being structured in a card index format, with each illusion being dynamically presented on a separate page and each theoretical chapter popping up when appropriate. Covered topics are grouped around the following categories: ambiguous figures (i.e., figure–ground segregation and geometric figures), paradoxes (i.e., in art), distortions (i.e., angle illusions and size constancy), fiction (i.e., illusory contours and consecutive effect), distance perception (i.e., static monocular cues, dynamic monocular cues, binocular cues), and miscellaneous topics (i.e., color perception, contrast, human eye, interpretation of shadow, organization of the perceptual field). Rapid access to any card is provided by a navigation menu.

As illustrated in Figure 1, in a typical illustration, students will see a picture accompanied by a short description of it. An icon representing a computer mouse is displayed at the bottom right side of the picture. By moving the mouse over it, the user will cause a modification of the picture, making clear that her/his percept was influenced by the illusion. The user gains access to the theoretical background associated with the illusion by clicking on the Category and/or Subcategory links displayed at the top of the picture. The explanation will pop up in a new window, allowing the reader to keep an eye on the picture while reading the text. Connections between the different illustrations and theory cards are made in a Section map page.

The Zöllner illusion presented here is categorized as a distortion; the perception is biased relative to reality. It is also subcategorized as an angle illusion: Many opticogeometrical illusions, such as the Zöllner, Hering, Poggendorf, and Müller-Lyer, can be explained by a tendency for the visual system to enlarge acute angles; lines that form an acute angle repel each other. The phenomenon could be due to purely optical factors that cause retinal fuzziness (spherical and chromatical abberation and diffraction) or to the phenomenon of lateral inhibition in contour detectors in cells of the retina and visual cortex.

Cognitive Processes Section

The Cognitive Processes Section is organized quite differently from the Illusions Section. In this section, readers are given a great deal of text presenting some of the most classical experimental paradigms. The theory is supplemented by a Java applet whereby people can participate in the associated experimental situation interactively. People are encouraged to first take the experiment as naïve participants on a separate page. However, they can also run the applet in a pop-up window from the theory page so that they can more easily grasp the different experimental conditions discussed in the text. Seven theoretical chapters are currently presented: aspects of a stimulus—features, dimensions, and configurations (Garner, 1974); cognitive evaluation (the *tower of Hanoi* problem); iconic memory (Sperling, 1960); interference and automaticity (Stroop, 1935); mental images (Shepard & Metzler, 1971); and neuropsychological testing (mirror drawing).

In a typical demonstration, a push-button version of the task is proposed, as illustrated in Figure 2 with the applet of the Stroop task (note that this task normally requires an oral response). On the top of the page (not visible in the figure), guidelines through the different experimental conditions are offered via a form to be filled in; the participant is asked to report his/her mean reaction times for each condition in order to receive feedback on performance. Instructions relative to the experiments are displayed on the applet. The experimental situation is displayed at the bottom of the page, and the experiment runs

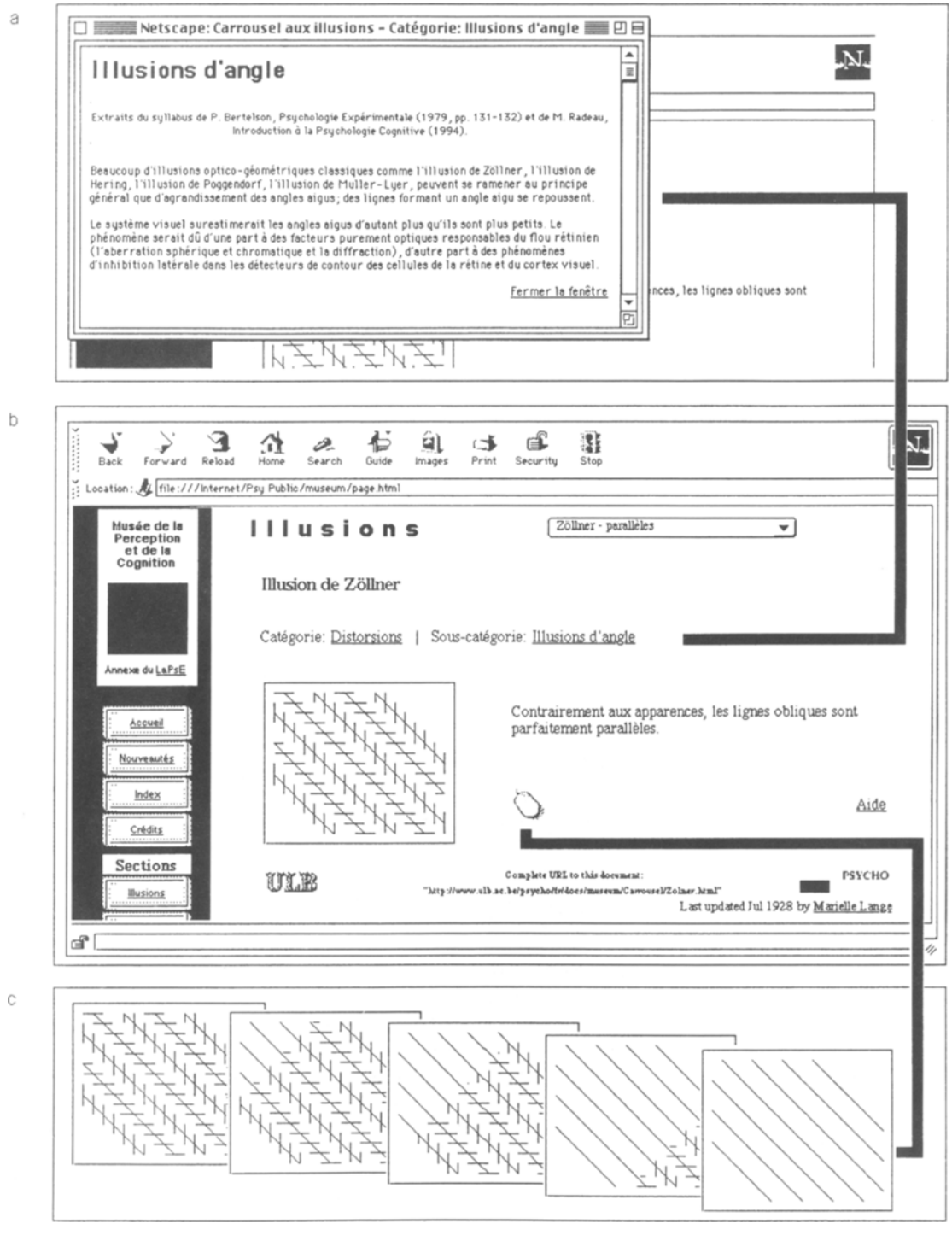

Figure 1. Interactive presentation of the Zöllner illusion in the Illusions Section of the Museum of Perception and Cognition website (text in French). The illusion is presented in a card index format (b). Clicking on the Category or Subcategory links will bring up the associated theoretical chapter (a), and moving the mouse over the mouse icon will start an image animation (c) and show that, contrary to what is perceived, the lines are in parallel.

as follows: The participant has to press one of four keys as soon as he/she identifies the color of the word appearing on the screen (the keys "C," "V," "B," and "N" for the colors red [rouge], green [vert], blue [bleu], and yellow [jaune], respectively). Reaction times are collected for each trial, and the mean reaction time for correct identifications is displayed after a block of 20 trials. Three conditions have been prepared: In the color-spots condition, the participants are presented with ink spots of one of the four colors; in the colored-words condition, colored mono-

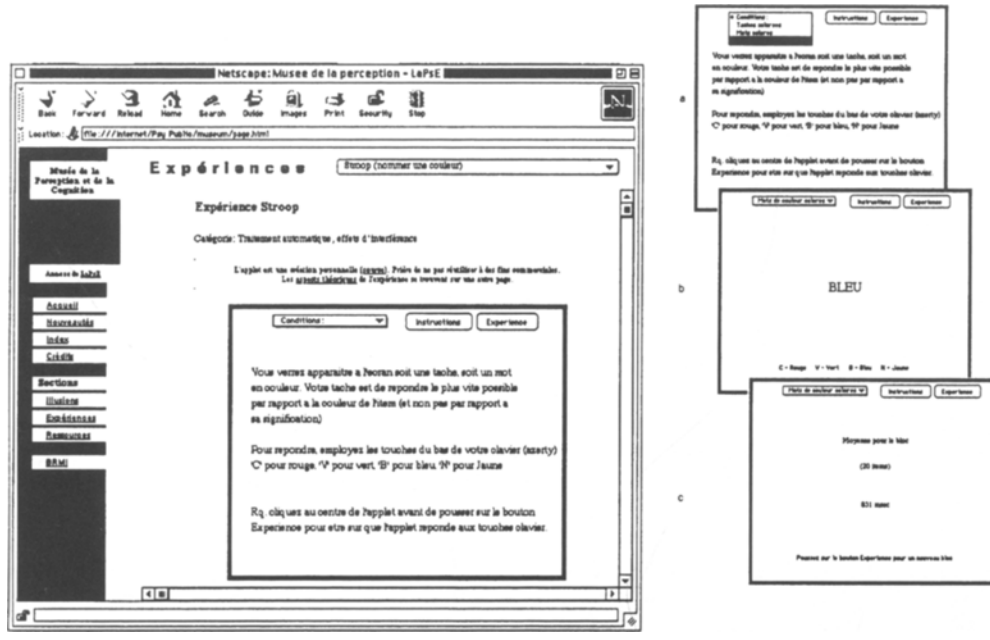

Figure 2. A Java-based lab demonstration in the Experiment Section of the Museum of Perception and Cognition website. The Stroop task Java applet is presented in the page context on the left; some representative moments are displayed on the right. These moments are as follows: (a) an experimental condition is being selected, and corresponding instructions are given. (b) The experiment is running, items appear at the center of the screen, and the user enters his/her response using predefined keyboard keys. (c) Reaction times are collected for each trial, and the mean reaction time for correct identifications is displayed after a block of 20 trials.

syllabic words are presented; and in the colored-color-names condition, words that represent colors (e.g., *red*, *blue*, and *yellow*) are presented with the words printed in a color different from the word's meaning (e.g., in Figure 2, the word *blue* is printed in red). As is well known, the Stroop effect experienced in this demonstration illustrates automaticity and interference. The classical result is that the reaction times in the colored-color-name condition are significantly longer than those in the colored-word condition. This is due to interference of the color identification by the processing of the word meaning.

UTILIZATION OF THE MUSEUM SITE IN A LECTURE CONTEXT

The museum site was built as the virtual extension of the Perception Museum set up by our lab but regrettably closed to the students for more than 2 years because of a lack of room and staff availability. The transfer of the museum to the Internet enabled us to reintroduce to the 1st-year students a session about perceptual processes associated with the experimental psychology lecture.

In place of the former museum visit, an Internet session is organized with students grouped in dyads. In agreement with the principles of interactivity and user control, this session privileges a computer-based rather than a computer-assisted teaching situation. Ideally, this enables students to discover the information at their own pace and to explore in depth or superficially various topics at their convenience, in accordance with their interest in the subject. Moreover, the Internet session provides an opportunity to introduce 1st-year students and novice computer users to the promising self-study tool constituted by the Net. One limitation of this approach is that it is impossible to predict how a reader or learner will interact with and process information in this computerized situation. This limitation is not serious, however, since these notions are presented to the students in the associated lecture, but it could pose a more significant problem if Internet sessions were intended to replace some of the lecture chapters. To ensure that students have adequately engaged with and understood the main contents of the session, dyads of students are given question sheets that they have to complete while exploring the Net-based material. A detailed analysis of students' responses, feedback, and session logs is then used to evaluate this new learning approach.

DEVELOPING JAVASCRIPT SKILLS FOR WEAVING A RICHER WEB

In this section, JavaScript features are introduced. JavaScript is a promising way for the information provider to add dynamism and interactivity. A set of examples prepared for this paper illustrates how JavaScript can be used to design more engaging, interactive, and effective Web-based presentations of course materials.

JavaScript Features

JavaScript is a programming language introduced by Sun in 1995. Like its parent, Java, it is an object-

oriented, multiplatform, and Internet-friendly computer language. An introduction to the JavaScript language can be found in Harriot (1997). Put briefly, JavaScript is an HTML-compliant language that permits lines of code to be pasted directly into Web pages. JavaScript is not the same as Java *scripts* (i.e., *applets*). Java *applets* are programs that depend on HTML for their distribution but that have been designed and precompiled using a specific software (e.g., JDK from Sun; an introduction to Java can be found in Briggs & Sheu, 1998). Java is a complete programming language that can be used for any Web-based application, including on-line experiments, lab demonstrations, and real-time simulations. JavaScript is not as powerful as Java. What JavaScript essentially provides is the possibility of designing documents that are updated automatically (e.g., color of the background, content of a form) or that react to user actions (e.g., mouse movement, button clicking). But JavaScript can do the latter in a large variety of ways. It has the ability to make dialogue boxes pop up, open new windows in the browser, and display messages to the user. As will be shown, these features can be used to liven up Web pages. Compared with other Web-compatible languages, JavaScript is characterized by four critical features: off-line utilization, ease of development, ease of use, and ease of learning.

No server required. JavaScript code is included in the Web document and interpreted by the browser after loading of the page. This presents two notable advantages. First, JavaScript can be used to control forms or image maps in exactly the same way as CGI scripts (programs that reside and are executed on the server), except that it has the ability to do it off line (i.e., in local browsing mode) as well as from a network-based connection. Thus, it allows authors of interactive Web documents to serve them over the Web as well as to distribute them on diskette or CD-ROM. Second, because all of JavaScript's processing is done by the client's computer, Web presentations that rely on JavaScript, rather than on CGI scripts, reduce network connections and decrease server load.

Ease of development. Contrary to the more elaborate Java or CGI scripts, no development tool or compiler is required; as for HTML, any text editor can be used to create JavaScript-enhanced documents. Any browser (i.e., Netscape Version 2.0 or later and Microsoft Internet Explorer Version 3.0 or later) can then be used to view the page.

Ease of use. Because many sites propose Cut&Paste libraries of scripts, beginners can start using JavaScript with no HTML skills beyond the ability to replace the existing text with text about their own subject. Furthermore, tools that automatically generate JavaScript are beginning to be available. For example, Web.builder.com [http://web.builder.com] proposes a valuable set of tools such as a quiz maker, a menu maker, and so on (links to these resources come with the on-line examples presented in the next section).

Ease of learning. JavaScript is referenced as an object-oriented language because the built-in variables are named as embedded objects (for instance, *document.bgColor* stands for the background color of the document). Besides this, JavaScript syntax is far more similar to languages such as BASIC or Pascal than to other object-oriented languages such as C++. Anybody already familiar with the World-Wide Web and the basics of creating Web pages with HTML will likely have little difficulty understanding the rudiments of JavaScript, even if he/she has no prior programming experience. Moreover, because of the popularity of the Web, he/she will benefit from a large variety of outside support (FAQs, discussion lists, tutorials, and on-line reference books).

JavaScript Potential

Basically, JavaScript acts upon HTML tags to modify some elements of the page (e.g., images, form elements, and frame content). It has, however, a limited range of actions. Three notable limitations are as follows: (1) JavaScript has no access to the text on a page in place (outside of a form or the *document.write* instruction, which necessitates a page reload); dHTML (dynamic HTML[2]) can be used to get around this. (2) Although JavaScript can manipulate data that the user enters in a form, it cannot send the resulting data back to the server; JavaScript has to be combined with CGI script to do this (Kieley, 1996). (3) JavaScript can flip images on the page, but it cannot create graphics—for example, it cannot draw a graph. Currently, Java is the best solution when you need to draw graphs.

JavaScript Basics

The only thing that is needed in order to run scripts written in JavaScript is a JavaScript-enabled browser—for example, the Netscape Navigator (Version 2.0 or later) or the Microsoft Internet Explorer (MSIE; Version 3.0 or later). JavaScript code is embedded directly into the HTML page. In order to show how this works, the following sections proceed through the easy example provided in the Appendix.

Script Execution

When the browser encounters the *document.write()* instruction placed in the head, it pauses momentarily to process the instruction, and "Hello!" is written on the very top of the page. The text "Fill in the text box and" present in the BODY part is then written on the page. The SCRIPT is processed, causing "push on the button below" to appear, and "Hello! Fill in the text box and push on the button below" appears on a single line. Next, an input box and a button are created with the normal <form> tag (HTML). When the button is pressed, the functions *alert1()* and *alert2()*, which were declared in the header, are executed. The first alert window then pops up, saying, "You pushed the button." This is followed by a second alert window, saying, "XXX was entered in the text box," where XXX is the text entered in the box (in uppercase).

Embedding JavaScript in HTML

As is apparent from this brief example, JavaScript commands are directly included within the HTML page. They can be found at three locations: (1) In the header

(inside the <HEAD> and </HEAD> tags at the top of the document) limited by the <SCRIPT> and </SCRIPT> tags. The <SCRIPT> tag is an extension to HTML that can enclose any number of JavaScript statements or functions. As attributes, it has LANGUAGE, which specifies the JavaScript version. To ensure that browsers that do not recognize JavaScript do not handle JavaScript code as text and write it on the screen, the JavaScript code should always be written within HTML comment tags (i.e., lines 4 and 20), with the ending comment tag preceded by a double slash. A script placed in the header will be run as the document loads. (2) JavaScript instructions can also be embedded in the body of the document using the same <SCRIPT> tag. The script included in the body is run while the page is being written and may affect the display of the page (lines 26–30). (3) JavaScript instructions can be found in the event handler for a specific part of the page. For example, the event handler onClick will call a JavaScript function when a button is pressed (see line 34).

Functions

JavaScript has some built-in functions. In the preceding example, *alert()*, *toUpperCase()*, and *write()* are all examples of built-in functions. JavaScript also allows user-defined functions, which are defined as follows:

```
function MyFunction(params) {
...function stuff...
}
```

MyFunction is the name of the function. All function names in a script must be unique, and a function must always be defined before its call; *params* consists of zero (empty parentheses are required when the function does not use parameters), one, or more parameter variables that you pass to the function. The variable name provided as the parameter in the function definition can then be used for processing within the function, as shown in line 12: The text to display in the alert window is passed as a parameter to the function. *Function stuff* is the instructions carried out by the function. The { and } brace characters define the function block and are absolutely necessary. Inside them you can put almost anything—variable definition, statements, or function calls. JavaScript supports a small collection of statements with some similarity to languages like C, including the usual *if*, *while*, and *for* statements.

Event Handlers

Event handlers enable a JavaScript instruction to be executed whenever an action is performed on a tagged element of the HTML document. Clicking a button and following a link are examples of some events. For instance, in line 34 a button is displayed and the onClick event *handler* is added in the <INPUT> HTML tag to tell JavaScript that when the form button is clicked, the JavaScript code specified should be processed. In our example, user-defined functions *alert1()* and *alert2()* are then called.

Event handlers can similarly be placed in anchor or body tags. As an illustration,

```
<A href="#" onMouseOver="alert('hello world')">
<img src="image.gif" border="0"></A>
```

will prompt a dialogue box on the screen showing "hello, world" when the mouse moves over the picture. And

```
<Body onUnLoad="alert('GoodBye')">Goodbye</A>
```

will prompt another dialogue box displaying "GoodBye" upon leaving the page.

Objects

JavaScript is an object-oriented language. This simply means that it can use objects. An object is a more complicated version of a variable. Rather than being assigned a value (e.g., $x = 2$), objects are assigned properties (e.g., *The color of my car is blue*) and methods (e.g., *Start the car*). These properties and methods are identified by naming the object followed by a point and its property (e.g., *myCar.color="blue"*) or method (e.g., *myCar.startEngine()*). In short, when you scrutinize or change the details about something that already exists, such as recovering the textBox value to assign it to the *alertMsg* variable (line 13), you are handling object properties. When you perform an action on an object, such as writing on the document (lines 6, 28), or convert a string to uppercase (line 14), you are using object methods—that is, functions associated with the object.

One unique feature of JavaScript compared with other object-oriented languages like C++ or Java is that it uses a simplified object model. In JavaScript, the user does not need to care about object declaration or variable typing when using built-in objects/variables of the type *Date*, *Math*, *String*, *Array*; as soon as a variable is used, it becomes an instance of the appropriate object class. For example, in line 13, *alertMsg="xxx"* defines a new instance of a string object without any instruction to do so.

In JavaScript, actual elements of the browser, such as windows, documents, frames, forms, links, and anchors, are represented as objects. For example, the term *document* used in lines 6 and 13 corresponds to a window-content object (the HTML document loaded in the browser) with specific properties (e.g., *forms[0].elements[0].value*) and methods (e.g., *write()*). In our example, there is one object in the document, a form and two objects in the form, a text field, and a button. The form could be referenced alternatively as *forms[0]*—first among all the forms contained in this document—and through *myForm* as specified with the NAME attribute in the FORM tag. In both cases, *myForm* has to be considered as an object's descendants and property of the document object and has to be accessed through it (line 13). Form elements are also stored in an elements array. As forms, they can be accessed by their position in the array—*elements[0]* stands for the text box and *elements[1]* for the button—or by their name. As an illustration, reference to the text

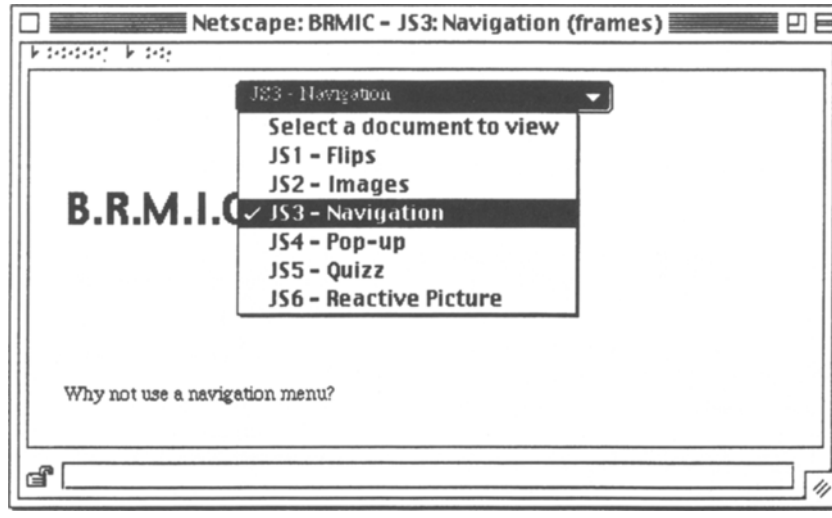

Figure 3. Navigation menu list.

field will be granted by *document.forms[0].elements[0]* or *document.myForm.myTextBox*.

ON-LINE EXAMPLES

The following examples were created to show how JavaScript can be used to design more interactive Web pages. Some of the scripts are used in our museum website, and others were developed for the present paper. This list of ideas is certainly not exhaustive; it merely represents some of the possible techniques. In addition, links to useful resources are provided in the on-line documents cited.

Examples can be tested interactively using the URL [http://www.ulb.ac.be/psycho/brmic.html]. For each example, a "view source" button is available. Pressing on it will cause the display of two files, the source code and an HTML document that proposes a step-by-step visualization of the JavaScript code used to implement the example. A package containing the different Web documents associated with the examples that are presented here can be obtained at the same address.[3] The idea behind these documents is to provide teachers with scripts they can easily adapt to their own needs, thereby encouraging further development of Web resources in psychology.

Example 1—Navigation Menu List

Rationale. Several studies suggest that the range and extent of user interaction with the content of a chapter increase as the user is given more freedom to navigate (Brown & Thompson, 1997; Megarry, 1989). On the Web, a pull-down menu is an easy way to provide fast access to any of the pages from anywhere in the document, thus making it easier for users to find their way around the site. It further facilitates rapid access to the needed location. Also, it is less space-consuming than a more traditional table of contents.

Description. The screen in Figure 3 is divided into two parts (i.e., two frames). A pull-down menu is displayed on the upper part of the screen. When the user clicks on the menu, the menu expands, proposing a list of documents to be viewed. The selected document will be loaded in the lower part of the screen. The presentation of this menu in a frame allows us to provide a static area that remains fixed as people visit portions of the site.

Utilization. A navigation menu helps to organize and make more transparent the structure of a website (or any section of the site). This menu is most appropriate for any presentation that does not follow a linear pathway. In some cases, it might also encourage the student to explore relations between concepts.

Example 2—Text and Image Flip

Rationale. Sometimes complex presentations are more thoroughly understood when the material is presented in discrete steps rather than in a perpetually looping animation.

Description. The script implemented in Example 2 gives the user control over the pace of the presentation. By clicking on dedicated controls (i.e., the three button-like icons standing for *next*, *previous*, and *first*; see Figure 4), the user can modify both the content of the text box and the image.

Utilization. This script is useful for presentation of complex materials that need to be presented one element at a time. For example, in a complex graph, one can present the axes and labels first, and then add each bar or line (Stoloff, 1995). In the Museum of Perception and Cognition website, a variant of the script is used to show the illusions in a dynamic way.

Example 3—Multi-Image Display

Rationale. A computer is a wonderful tool for displaying image-based presentations but a poor medium to be used as a page turner. Long documents are often hard

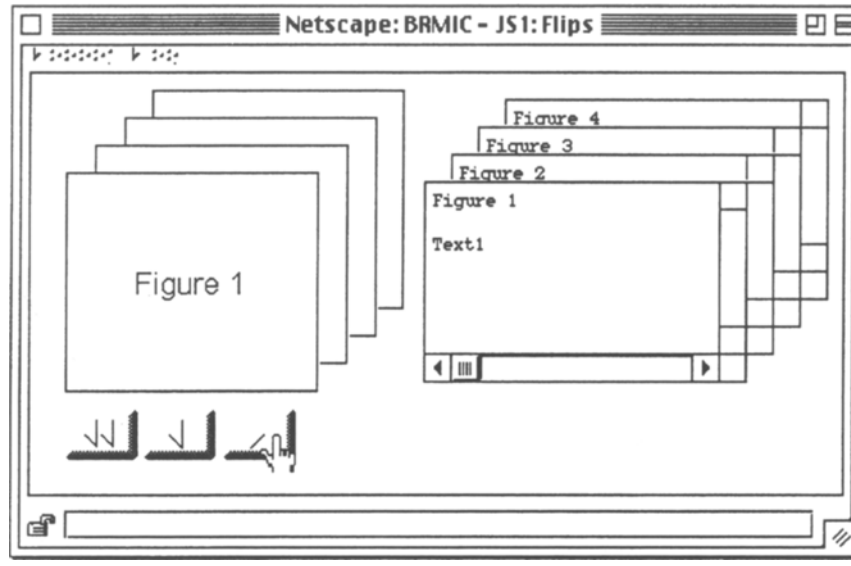

Figure 4. Text and image flip.

to navigate or peruse for an overview. This script helps authors to reduce the length, but not the content, of the presentation (and contributes to more engagement from the user).

Description. Example 3 is a script that displays several pictures at the same location (Figure 5). If the user moves the mouse over one of three little squares appearing at the right of each subtitle, the corresponding picture will be presented in the display area (top, middle, and bottom squares will, respectively, display pictures corresponding to Rubin, Rabbit-duck, and Boring).

Utilization. As this example suggests, information in a multimedia world can be presented in new and different ways. This script could be included in computer-based lectures or book presentations that involve a large number of pictures.

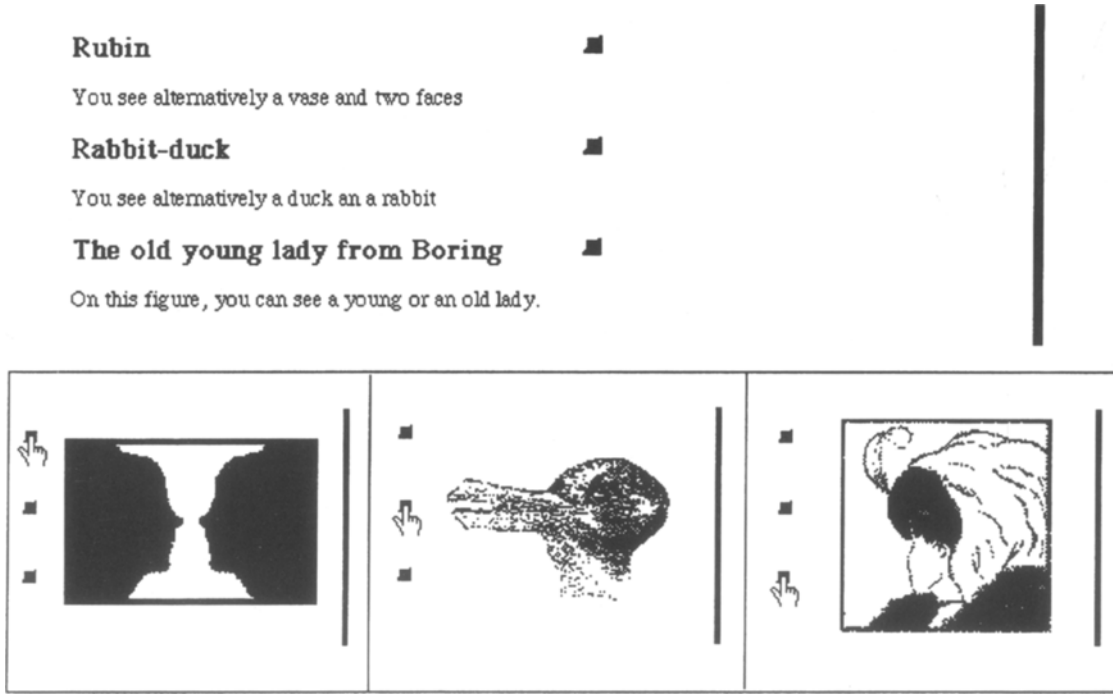

Figure 5. Multi-image display.

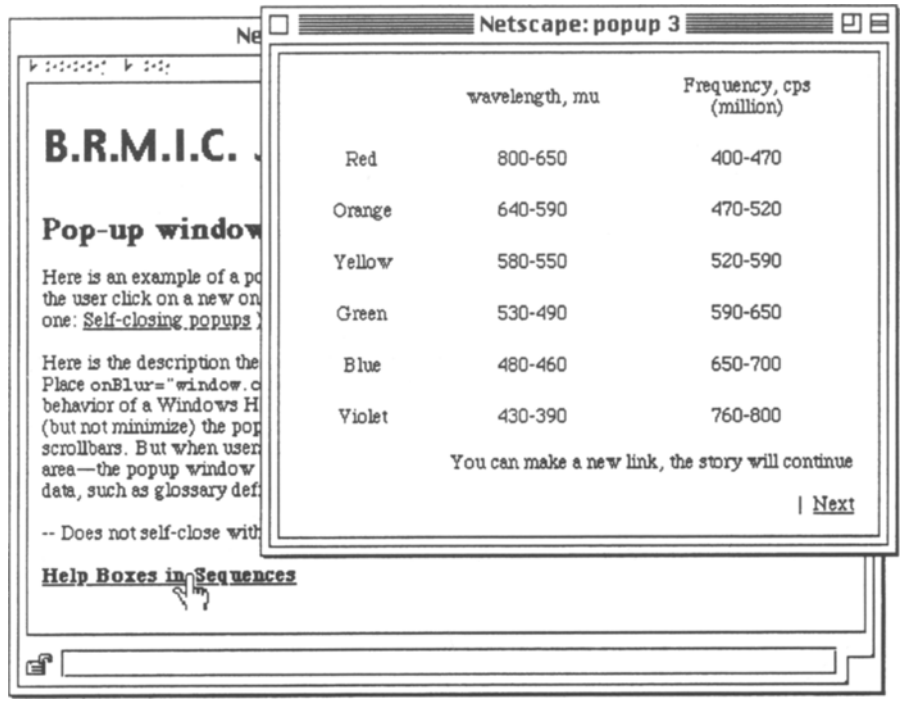

Figure 6. Pop-up window.

Example 4—Pop-up Window

Rationale. Because not all users have the same level of proficiency, it might be important to give them the opportunity to skip material that has already been mastered or that is of no interest to them. Pop-up windows provide a simple way to allow the reader to tailor his/her readings.

Description. When the user clicks on the Help Boxes in Sequence link on the bottom of the page (Figure 6), a pop-up window appears on the screen. Some lines of code have been added so that the user does not need to keep closing the window. He/she can just click anywhere out of the pop-up window (e.g., on the main document) to make it vanish.

Utilization. Pop-up windows are a method of presenting information in which selected words from the text can be expanded at any time, providing other information about the word. This expansion is usually provided via highlighted links, but could be provided by other HTML elements as well (form button, picture, reactive maps, etc.). They can be used anywhere to provide the opportunity to access additional information on a topic.

Example 5—Quiz

Rationale. Following Ramsden (1985), assessment has a major influence on how and what students learn. Furthermore, "good responsive feedback from quizzes is highly motivating as it helps students discover what it is they know" (Cox & Clubb, 1995).

Description. Figure 7 shows an HTML form containing a simple multiple choice question to be answered. When the user clicks on the correct response (i.e., *toggle* "*1953*"), an alert box showing "Correct!" appears. Two links are also provided. They both cause an alert box to show up (obviously, the Hint link proposes a clue to the response and the Answer link displays the solution). A more elaborate example of the way JavaScript can interact with forms, as well as a form gallery, is available on line.

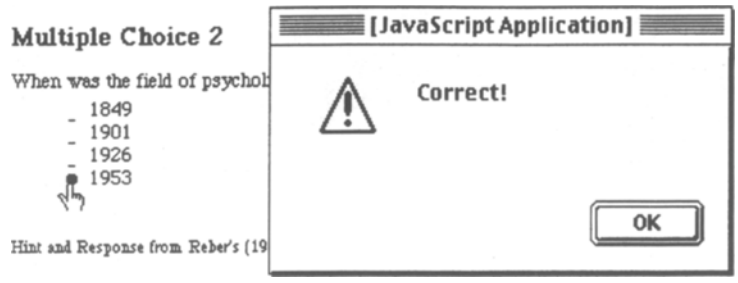

Figure 7. Quiz.

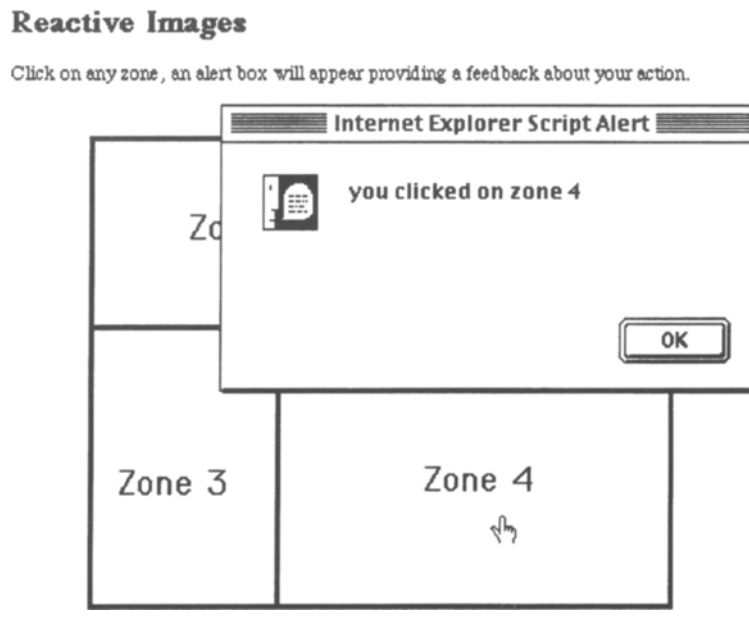

Figure 8. Reactive picture.

Utilization. Forms in effect allow users to input data by clicking on check boxes, radio buttons, menus, buttons, and also by typing into text fields. A quiz can easily be implemented using Form tags; scripts can then be used to check responses and react in variable ways to the user's input. In the context of a tutorial presentation, quizzes can be used everywhere for prompting responses from the user; these can be evaluated and either corrected or signaled as correct. If the computer is connected to a server, the results of each user can be sent to a machine using CGI protocol (e.g., for on-line examinations) or e-mailed to the student or the teacher. This enables later analysis of the appropriateness of the questions—information that could be of interest to both students and teachers.

Example 6—Reactive Picture

Rationale. Viewing a map of the land or a spatial layout may provide an understanding of problems that is not evident in a traditional text-based description. Moreover, it is important to offer flexibility in the presentation for different types of learners. Explanations can sometimes be favorably supplemented by a clickable picture.

Description. A simple graph is displayed at the center of the screen (Figure 8). It is divided into four zones (Zones 1 to 4). When the user clicks on any of the zones, the comment "You clicked on zone x" will appear in a pop-up dialogue box. In the on-line version, a more elaborate example simulates a visual search experiment (i.e., *Find a specific letter in a matrix of letters*).

Utilization. Image maps enable an image to be sensitized so that clicking in a particular region will cause an action. The action to be carried out could be the loading of another document, the popping up of a dialogue box, or the opening of a new window in the browser. Reactive maps can be used effectively in many different situations: Such maps can be used to display detailed information on graphs (e.g., when the user moves the cursor over a chart, a detailed description could appear in a pop-up dialogue box), to organize the navigation (Mashinter & Kraiker, 1997), or to ask the user to respond to a visual display in quiz-like questions (Krantz & Eagley, 1996).

CONCLUSION

The first part of this paper described an on-line tutorial, the Museum of Perception and Cognition, the aim of which is to help students revise lecture materials at home, but also to motivate them, to encourage them to learn by themselves, and to encourage them to consult external sources. Starting from this concrete realization, the second part of the paper introduced the candidate Web lecturer to the JavaScript language, a resourceful, though easy to use, feature of the Net that can be used to add interactivity to on-line course presentations.

REFERENCES

BRIGGS, N. E., & SHEU, C. F. (1998). Using Java in introductory statistics. *Behavior Research Methods, Instruments, & Computers*, **30**, 246-249.

BROWN, A., & THOMPSON, H. (1997). Course design for the WWW—Keeping online students onside. *ASCILITE'97 Conference Proceedings* [On line]. Perth, Australia. Available: http://www.curtin.edu.au/conference/ASCILITE97/papers/Brown/Brown.html [1998, July 7]

COX, K., & CLUBB, O. (1995). Formative quizzes and the World-Wide Web. *ASCILITE'95 Conference Proceedings* [On line]. Melbourne, Australia. Available: http://ASCILITE95.unimelb.edu.au/SMTU/ASCILITE95/abstracts/Cox.html [1998, July 7]

GARNER, W. R. (1974). *The processing of information and structure*. Hillsdale, NJ: Erlbaum.

GAZZARD, S., & DALZIEL, J. (1997). How it looks from their side of the screen: Student evaluations of a World-Wide Web tutorial in the De-

partment of Psychology at the University of Sydney. *ASCILITE'97 Conference Proceedings* [On line]. Perth, Australia. Available: http://www.curtin.edu.au/conference/ASCILITE97/papers/Gazzard/Gazzard.html [1998, July 7]

HARRIOT, J. S. (1997). Using JavaScript to build a psychology practice exam. *Behavior Research Methods, Instruments, & Computers*, **29**, 232-236.

KENNEDY, D. M., & MCNAUGHT, C. (1996). Interactive multimedia: Educational desert or educational oasis? *ASCILITE'96 Conference Proceedings* [On line]. Adelaide, Australia. Available: http://www.ascilite.org.au/conf96/12.html [1998, July 7]

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

KRANTZ, J. H., & EAGLEY, B. M. (1996). Creating psychological tutorials on the World-Wide Web. *Behavior Research Methods, Instruments, & Computers*, **28**, 156-160.

LAURILLARD, D. (1993). *Rethinking university teaching: A framework for the effective use of educational technology*. London: Routledge.

MASHINTER, G., & KRAIKER, R. (1997). Delivering courseware via a CD-ROM website. *ASCILITE'97 Conference Proceedings* [On line]. Perth, Australia. Available: http://www.curtin.edu.au/conference/ASCILITE97/papers/Mashinter/Mashinter.html [1998, July 7]

MEGARRY, J. (1989). Hypertext and compact discs: The challenge of multimedia learning. In C. Bell, J. Davies, & R. Winders (Eds.), *Promoting learning: Aspects of educational and training technology XXII*. London: Kogan Page.

O'REILLY, M., & MORGAN, C. (1997). Designing WebCDs: A low cost option to enhance learning and interaction. *ASCILITE'97 Conference Proceedings* [On line]. Perth, Australia. Available: http://www.curtin.edu.au/conference/ASCILITE97/papers/O'reilly/O'reilly.html [1998, July 7]

PETERSON, N. K., & ORDE, B. J. (1995). Implementing multimedia in the middle school curriculum: Pros, cons, and lessons learned. *T.H.E. Journal*, **22**, 70-75.

RAMSDEN, P. (1985). Student learning research: Retrospect and prospect. *Higher Education Research & Development*, **4**, 51-69.

RAMSDEN, P. (1992). *Learning to teach in higher education*. London: Routledge.

SHEPARD, R. N., & METZLER, J. (1971). Mental rotation of three-dimensional objects. *Science*, **171**, 701-703.

SIMS, R., & HEDBERG, J. (1995). Dimensions of learner control. A reappraisal for interactive multimedia instruction. *ASCILITE'95 Conference Proceedings* [On line]. Melbourne, Australia. Available: http://ASCILITE95.unimelb.edu.au/SMTU/ASCILITE95/abstracts/ Sims.html [1998, July 7]

SPERLING, G. (1960). The information available in brief visual presentations. *Psychological Monographs*, **74** (Whole No. 498), 1-29.

STOLOFF, M. (1995). Teaching physiological psychology in a multimedia classroom. *Teaching of Psychology*, **22**, 138-141.

STROOP, J. R. (1935). Studies of interference in serial verbal reactions. *Journal of Experimental Psychology*, **18**, 643-662.

NOTES

1. Documents presented in this website are all in French, but an abridged English version of the site has been prepared for the reader of this paper. The English pages can be accessed by clicking on the English flag on the homepage or on the website menu.

2. dHTML (dynamic HTML) is a recent extension of JavaScript that further allows one to dynamically change the text displayed on a page or move some parts of the screen. Features of dHTML are not discussed in this paper. Links to dHTML resources and tutorials are provided in on-line documents.

3. Lab demonstrations implemented in Java can be viewed in the museum website, and a package containing sources of Java applets presented in the museum website can be obtained on the BRMIC page of the site—URL [http://www.ulb.ac.be/psycho/brmic.html]. These Java programs can be easily modified to present other demonstrations.

APPENDIX

Listing 1: A simple JavaScript

```
<HTML>
<HEAD> <TITLE>JavaScript Example</TITLE>
<SCRIPT LANGUAGE="JavaScript">
<!— Begin hiding form older browsers

document.write("Hello!")

function alert1() {
    alert("You pushed on the button!");
}

function alert2(str) {
    alertMsg = document.forms[0].elements[0].value
    alertMsg = alertMsg.toUpperCase()
    alertMsg += " was entered in the "
    alertMsg += str
    alert(alertMsg);
}

// End hiding —>
</SCRIPT>
</HEAD>
<BODY>

Fill in the text box and
<SCRIPT LANGUAGE="JavaScript">
<!— Begin hiding form older browsers
```

APPENDIX (Continued)

```
28        document.write(" push on the button below")
29    // End hiding —>
30    </SCRIPT>
31
32    <FORM NAME="myForm">
33    <INPUT TYPE="Text" NAME="myTextBox" VALUE="Type something">
34    <INPUT TYPE="button" onClick="alert1();alert2('Text box')" VALUE="Push me">
35    </FORM>
36
37    </BODY>
38    </HTML>
```

(Manuscript received August 17, 1998;
revision accepted November 6, 1998.)