JavaScript in Language Teaching Web Sites

Martin Beaudoin
Faculté Saint-Jean, University of Alberta

"The diversity of functions that can be written with JavaScript is limited only by your imagination."

The Web is interesting for language teaching because it has the potential to integrate input by learners in distance learning contexts. Furthermore, it is based on a language (HTML) that is very simple. However, HTML is not very powerful and has not been built for educational purposes. Various attempts to add interactivity to the Web have been made. One of the most successful ones is a scripting language called JavaScript. Originally created by Netscape, it now works on *Internet Explorer*, although not always as successfully. JavaScript is a powerful language that can be inserted within a webpage. This is especially useful because it is faster than accessing a server every time information needs to be processed. The scripts are generally small, thus making very little difference for loading time. The diversity of functions that can be written with JavaScript is limited only by your imagination. Aside from the main limitation of the language (it does not allow reading or writing files on the user's machine), almost anything is possible.

This flexibility is the main asset of JavaScript. It means that a script can provide information to the learner as he/she reads documents, verifies answers given to an exercise, compiles results and sends information back to the instructor. It can also be used to create exercises that will adapt to the learner's responses, provide contextualized help, create collapsible tables of contents, tabulate marks etc. This article will present some of the ways I use JavaScript in a large French grammar web site that I am developing. Not all of these functions can be shown here because of a lack of space. Most scripts can be found in action in Beaudoin (1997a). I will assume basic knowledge of JavaScript and HTML. However, if you do not have such basic knowledge, please continue reading because the language is simple, and most users start by cutting and pasting functions.

Contextualized Help

People can get lost visiting sites even if the directions and the introduction to a web site are crystal clear, because people often do not read the information. They generally prefer to find information as they need it. Two alternative routes to providing information are through the status bar (at the bottom of the screen) and through a message box. For example, a description of an icon's behavior can appear in the status bar as the mouse passes over it, or a message box can be presented when a page is opened for the first time by the user. These functions are initiated by different portions of a webpage. The former function is called by the anchor point, whereas the latter is called within the <BODY> tag. This message box will appear every time the page is loaded or refreshed. The script can be modified to show the message only the first time the page is loaded.

Status bar text:

```
        <A HREF="First.htm" TARGET="_parent"
onMouseOver="window.status='This brings you to the first page of the site.'
; return true;"> <IMG BORDER=0 SRC="up.gif"></A>
```

Message box:

```
        <BODY onLoad="alert ('Welcome to my first web page.')">
```

Searching Lexicons and Texts

If you present long lists of vocabulary where lexical entries are defined on other pages through links, a mini search-engine may be an asset to your page. A CGI would probably be more efficient than a script for this purpose, but it requires server access, which is sometimes difficult for people with bad or slow connections. The following script is used to show the definition for a word taken from a list of words in another frame (the definition is contained in an HTML document that includes all definitions). It requires two hidden variables to be placed around the lexical entry for which you want to make a search possible. The first one is the actual word to be searched and the second one is the targeted anchor point for this particular word. This script can be seen in action in Beaudoin (1997b).

```
<INPUT TYPE="hidden" NAME="word" VALUE="lueur">
<INPUT TYPE="hidden" NAME="point" VALUE="definition.htm#lueur">
```

Search function:

```
function Search(){
        Value = document.form.search.value;
        var q=1;
        while (q <= document.form.word.length){
            p = q-1;
            q++ ;
```

```
                Word = document.form.word[p].value;
                Point = document.form.point[p].value
                if (Word.indexOf(Value) >= 0) {
                        parent.frames[1].location.href = Point;
                        return;
                }
            }
}
```

Automatic Correction

One common problem observed in language teaching on the Web is that the answers to exercises are often found on different pages. In this way, the learner cannot compare his/her answers with the correct one. This is especially disruptive when the learners have to provide written answers. The following script will correct multiple choice or fill-in-the-blank questions (specific scripts exist elsewhere). It can also easily be modified for multiple answers. It requires a hidden variable containing the correct answer(s) for each question or problem of the exercise. The score returned to the learner is on the number of questions answered. This script can be seen at work in Beaudoin (1997c). Separate scripts for multiple choice questions and fill-in-the-blanks also exist (Beaudoin 1997d; Beaudoin 1997e).

Automatic correction function:

```
function Correct(form) {
// option part
        note = count = total = 0;
        while (count != form.Opt.length){
            if (form.Opt[count].selectedIndex == form.Optc[count].value){
                        note++;
                        total++;
                        form.Optb[count].checked = false;
            }else{
                        if (form.Opt[count].selectedIndex != 0) total++;
                        form.Opt[count].selectedIndex =
form.Optc[count].value;
                        form.Optb[count].checked = true;
            }
            form.result.value = (note + " / " + total);
            count++;
        }
// fill-in-the-blank part
            count = 0;
            while (form.Fill[count].value != ""){
                        if (form.Fill[count].value == form.Fillb[count].value){
                                    note++;
                                    total++;
            }else{
                        form.Fill[count].value = (form.Fillb[count].value + " *"
+
                        form.Fill[count].value + "*");
                        total++;
                }
                count++;
                form.result.value = (note + " / " + total);
                if (count== form.Fill.length) break;
            }
//The following fills unanswered questions (can be commented out)
            while (count != form.Fill.length){
                form.Fill[count].value = (form.Fillb[count].value + " **");
                count++;
            }
}
```

Scoring and Tabulation of Marks

I do not tabulate marks for exercises because doing so would emphasize the results instead of the process, which I believe is most important in language learning. I can see, however, that it can be useful in some circumstances. The results achieved by the learner can be recorded for later reference or sent to the instructor. They require the use of cookies, a semi-permanent writing of data on the user's hard-drive. Record keeping of the learner's scores involves two steps: (a) adding up all of the scores and storing them; and (b) retrieving the information. The cookie stores the number of exercises completed, the total number of points earned (the number of correct answers supplied by the student), and the maximum number of points attainable (the perfect score). There are two ways of retrieving this information (one by the student, and one by the instructor), each of which requires a specific function. Furthermore, you may want to retrieve this information for conditional redirection of the learner (this will be shown later). Retrieval by the student involves the display of the results in a message box stating "Your total score for the last [number of exercises] exercises is [score] on a maximum of [maximum]. This means that you achieved [score in %]." The instructor's method requires that the information be transmitted through e-mail.

"I do not tabulate marks for exercises because doing so would emphasize the results instead of the process, which I believe is most important in language learning."

The function illustrated below is to add and save the marks (AddMark). It is important to note that it requires the basic cookie functions written by Bill Dortch (1996). To use the AddMark function, you simply have to put the following line at the end of your correction script: "AddMark (NewScore, NewDiv);".

Mark tabulation function:

```
//Insert cookie function written by Bill Dortch here.
function AddMark (NewScore, NewDiv) {
        var expdate = new Date ();
        expdate.setTime (expdate.getTime() + 31536000000);
/******************
*Gets cookie "mark" with old mark:
*       1st element = number of exercises so far
*       2nd element = the score so far
*       3rd element = what the score is on
******************/
        Mark = (GetCookie("mark"));
        if (Mark == null) Mark = "0;0;0";
        a = new Array ();
        a = Mark.split(";");
        a[0]++;
        a[1] = parseInt (a[1]) + NewScore;      //Adds new score to old one
        a[2] = parseInt (a[2]) + NewDiv;        //Adds new total to old one
        Mark = a[0] + ";" + a[1] + ";" + a[2];
        SetCookie ("mark", Mark, expdate, "/");    //Sets "mark" for 1 year
}
```

"It is preferable, in my opinion, to leave the decision to the learner by asking him/her to send the scores if he/she chooses to."

The student's retrieval function can be called for by a submit button using the following HTML line in a form:

```
<INPUT TYPE="BUTTON" VALUE="Show results" OnClick="PostMark()">
```

Student's retrieval function:

```
//Insert cookie function written by Bill Dortch here.
function PostMark () { //Shows the results to the learner
        var expdate = new Date ();
        expdate.setTime (expdate.getTime() + 31536000000);
        Mark = (GetCookie("mark"));
        a = new Array ();
        a = Mark.split(";");
        var percent = ""
        if (a[1] != 0) percent = "\nThis means " + ((a[1] * 100) / a[2]) + "%.";
        if (a[0] != 0)alert ("Your total score for the last " + a[0] + " exercises
is" + a[1] + " on a maximum of " + a[2] + "." + percent);
        Mark = "0;0;0";
        SetCookie ("mark", Mark, expdate, "/");      //Resets cookie "mark".
}
```

Sending the scoring information to the instructor may be important to help him/her keep track of the progress of the learner. It is preferable, in my opinion, to leave the decision to the learner by asking him/her to send the scores if he/she chooses to. The instructor's retrieval function involves sending the information through email. You thus need to call up the function that retrieves the information from the cookie, and then send it to the instructor. For security reasons, JavaScript does not allow you to send an email message silently. You should first make a page for the "SendMark ()" function. This function is called when the page is loaded up. Instead of sending the score to your personal email address, you should consider sending it first to a mail CGI that parses what is sent. Contact your webmaster for more information on this topic.

The following portion of HTML must be present in the webpage:

```
<BODY onLoad="SendMark()">
<FORM ACTION="mailto:your_address@your.mail.server"
METHOD="POST">
<INPUT TYPE="hidden" NAME="score" VALUE="">
<INPUT TYPE="submit" VALUE="Send score">
</FORM>
```

Instructor's retrieval function:

```
//Insert cookie function written by Bill Dortch here.
function SendMark (form) {
        NowDate = new Date ();
        Mark = (GetCookie(“mark”));
        a = new Array ();
        a = Mark.split(“;”);
        document.form.score.value = NowDate + “: The total score
for the last ”
+ a[0] + “ exercises is ” + a[1] + “ on a maximum of ” + a[2] + “.\n”
+ “This
means ” + ((a[1] * 100) / a[2]) + “%.”;
        Mark = “0;0;0”;
        SetCookie (“mark”, Mark, expdate, “/”);        //Resets
cookie “mark”.
}
```

Retrieving the score information and using it for conditional redirection is very simple. The following script reads the score, decides if it meets a preset standard, and directs the learner to a webpage that should best meet his/ her needs. A message could be posted to inform the learner of the process and the course of events. This script needs to be adapted to your specific needs before it will work.

Conditional redirection function:

```
function ConditionalTransfer (form) {
        Mark = (GetCookie(“mark”));
        PresetValue = “”;               //Fill-in minimal score in
%.
        URL1 = “”;            //Fill-in URLs for successful
students.
        URL2 = “”;            //Fill-in URLs for unsuccessful
students.
        a = new Array ();
        a = Mark.split(“;”);
```

```
    Score = ((a[1] * 100) / a[2]);
    ((Score >= PresetValue) ? location = URL1 : location =
URL2);
//Uncomment following line to post message to learner. Adapt
message to your needs.
    //alert ("Your score is " + Score + "%. This score is " +
    ((Score >= PresetValue) ? PresetValue : "not ") +
"satisfactory.");
}
```

Collapsible Table of Contents

Long tables of contents (TOC) are difficult to read, especially if you are interested in only a portion of the document. This is where a collapsible TOC becomes useful. In such a TOC, only the first level initially appears to the readers. They then have the option to click on the element they want to explore to itemize its contents. This second level can also be itemized if it contains sub-levels. The program that manages a collapsible TOC is simple but it has to be placed in a second frame (I set the size of this frame to "0"). However easy it is, setting the values of the menus takes time and space. A label, the number of elements it comprises, the initial state of the element and an optional action have to be set for each element of the TOC. The script allows a maximum of three levels of TOC. It can easily be increased. The script can be seen at work in Beaudoin (1997f).

Verb Conjugation

French has a complicated verbal system. A minimum of eighty-four templates are required to cover verb endings. Learners of French (L1 and L2) often need to look up verb endings in references. My web site incorporates a verb conjugation module that provides them for all of the thousands of French verbs. The verb module contains two frames: on the left one, the user enters the verb for which the template is desired and selects the verb mode and tense. The appropriate template is shown on the right frame at the given mode and tense. The user can also navigate through the other modes and tenses for that template. Suggestions for replacing rare or obsolete verbs are often available. This script is not shown here because it is too long but it can be found at work in Beaudoin (1997h), and the full script can be found in Beaudoin (1997i).

Conclusion

JavaScript adds functionality to webpages. The functions presented here are only a few of the many that can be used in language teaching (Beaudoin 1997a includes even more scripts). Limiting your self to HTML for language teaching on the Web defeats the purpose of transferring from a printed format to an online resource because it then is nothing more than an online book with little or no functionality and interactivity. JavaScript, and other scripting languages for that purpose, allows language teachers to provide a rich and effective learning environment for the learners.◆

Works Cited

Beaudoin, M. FRANC 161. (1997a) **http://www.atl.ualberta.ca/griff/A.htm.**

Beaudoin, M. Index. (1997b) **http://www.atl.ualberta.ca/griff/i.htm.**

Beaudoin, M. Français 161: exercice 4-11-36. (1997c) **http://www.atl.ualberta.ca/griff/4-11-36.htm.**

Beaudoin, M. Français 161: exercice 2-7-5. (1997d) **http://www.atl.ualberta.ca/griff/2-7-5.htm.**

Beaudoin, M. Français 161: exercice 1-11-10. (1997e) **http://www.atl.ualberta.ca/griff/1-11-10.htm.**

Beaudoin, M. Collapsible table of content. (1997f) **http://www.ualberta.ca/~mbeaudoi/menu.htm.**

Beaudoin, M. Script for a collapsible TOC. (1997g) **http://www.ualberta.ca/~mbeaudoi/collapsible.htm.**

Beaudoin, M. Conjugaison. (1997h) **http://www.atl.ualberta.ca/griff/conj.htm.**

Beaudoin, M. Script for verbal conjugation. (1997i) **http://www.ualberta.ca/~mbeaudoi/verbescr.htm.**

Dortch, B. Cookie.txt. (1996) **http://www.hidaho.com/cookies/cookie.txt.**

Martin Beaudoin is a professor of linguistics at Faculté Saint-Jean (University of Alberta). He graduated with a Ph.D. in Romance Linguistics from the University of Alberta in 1996. He teaches linguistics, French and CALL.