Interactive learning aided by JavaScript

A. Wise
The Robert Gordon University

This paper presents a case study in which some of the features of JavaScript have been employed to support the learning environment of students. Students have access to notes, self-assessment tests, and revision crossword puzzles. JavaScript is sufficiently advanced to permit the writing of a simple nutritional analysis program. However, there are some problems caused by slight incompatibilities between browsers, but this complication is of no importance when students have access only to one browser on the network.

Introduction

In recent years, the use of information technology to support learning in nutrition education has moved from mainframe-based programs to PC and Internet-based systems (Wise, 1986, 1998). Many lecturers are now exploring the use of the Internet and Intranets for education and this adds a new potential method for providing support to students. A search of the Internet for a technical term frequently links to pages containing notes for students. Some of these pages contain interactive elements that test whether students have learned the material. This paper is a case study that attempts to show how simple it is to build interactivity for students using JavaScript. Many programs can be written using the language Java, but the browser needs to be specially configured and considerable programming experience is required. Java is frequently confused with JavaScript, which is actually a language supported entirely within the browser itself. A browser will interpret code in JavaScript that is simply included within appropriate HTML tags on the page. JavaScript is used to provide simple interactivity and most people will have spotted its common features: buttons and text entry used on forms to provide feedback to the originators of Web pages. An advantage of JavaScript is that browsers are able to use this language, whatever the operating system, not only in Windows.

This paper describes the use of JavaScript in developing learning-support materials for students and shows that, whilst simple, JavaScript is able to tackle some relatively complex

tasks. The code for some steps in the programs is given to encourage people to try JavaScript; a book by Flanagan (1998) is all that is required to master this language.

It should be noted that there are some incompatibilities between different dialects of JavaScript. Since our students of nutrition currently have access only to Netscape 3, all the programs used for them have been written in JavaScript 1.1, which runs in that browser. Almost all the programs described here as being currently used by students will also work on Internet Explorer 3. The last two programs to be described (crossword puzzles and 'WebDiets') were written in JavaScript 1.2; 'Webdiets' operates in Netscape 4 and Internet Explorer 4. A less elegant version of the crossword program is currently working in both Netscape 3 and Internet Explorer 4. It is important to note that when such complex programs are to be used on the Internet they need code to detect what type of browser is being used and hence read the code designed for that browser. Some knowledge of the structure of a page of HTML is assumed in this discussion.

Placing a form containing interactive elements on the page

Figure 1 shows part of the page containing four 'radio' buttons and a 'select' list. The features of the opening screen for students demonstrate how simple it is to set up tags containing elements that react to user input. Table 1(a) shows the HTML code setting up and naming a form, followed by four 'radio' buttons, each with the same name. Then follows the 'select' list, with each of the ten options that can be used to fill the list box; all are blank except the last, which has a row of full stops to provide a box of the required size (see Table 1(a)). The code for buttons on the screen will be discussed later, but it should be pointed out that after the code for them, the form is closed (</form>), but this step is not shown in the example. The idea of the opening screen is to present students with choices depending on their year of study and then, within a year, the actual module being studied.

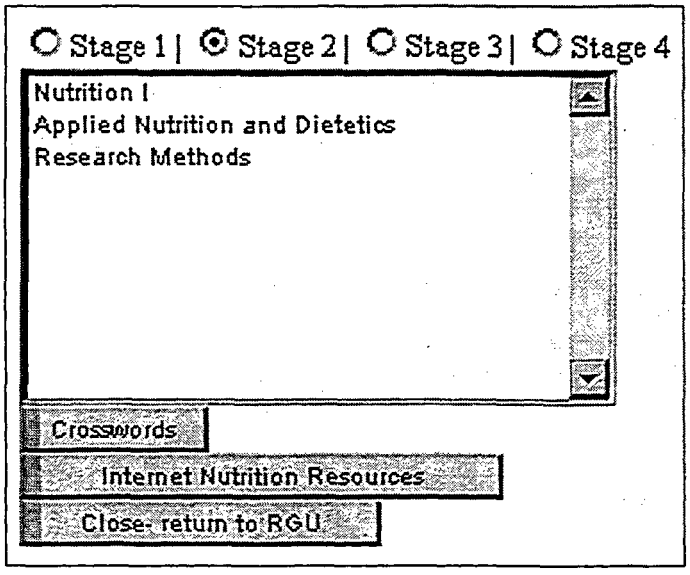

Figure 1: Part of the opening screen.

```
a)
<form name="firstform">
<input type="radio"  name="s" onclick="loadm(1)">Stage 1 |
<input type="radio"  name="s" onclick="loadm(2)">Stage 2 |
<input type="radio"  name="s" onclick="loadm(3)">Stage 3 |
<input type="radio"  name="s" onclick="loadm(4)">Stage 4<br>
<select name="m" size=10 multiple onchange="whensel()">
<option><option><option><option><option><option><option><option><option><option>.....
.....
</select>
b)
function fs()
{document.firstform.m.options[9].text="";}
</script>
</head>
<body text="#000000" bgcolor="#ffffff" background="wave.gif" onload="fs()">
c)
function loadm(which)
{document.firstform.m.selectedIndex=-1;
for(count=0;count<10;count++)
{document.firstform.m.options[count].text=""}
if(which==1)
{document.firstform.m.options[0].text="Social and International Nutrition";}
if(which==2)
{document.firstform.m.options[0].text="Nutrition I";
document.firstform.m.options[1].text="Applied Nutrition and Dietetics";
document.firstform.m.options[2].text="Research Methods";}
if(which==3)
{document.firstform.m.options[0].text="Nutrition II and III";}
if(which==4)
{document.firstform.m.options[0].text="Advanced Nutrition";}}
d)
if(document.firstform.s[1].checked==true)
{if(document.firstform.m.selectedIndex==0)fil="nut1/nut1.htm";
if(document.firstform.m.selectedIndex==1)fil="appnut/appnut.htm";
if(document.firstform.m.selectedIndex==2){xx=window.open('resmeth/res-
des.htm','newwin','scrollbars=no');fil="";}}
e)
{xx=window.open('','newwin','scrollbars=yes')};
xx.document.open();
xx.document.write('<head><title>Notes and Internet Resources</title></head>');
xx.document.write('<frameset cols="25%,*">');
xx.document.write('<frame src="' + fil + '" name="frame1">');
xx.document.write('<frame src="blank.htm" name="frame2">');
xx.document.write('</frameset>');
xx.document.close()}
```

Table 1: Code used in the opening screen.

Functions responding to window events

All the code for what the program is to do when elements are clicked is generally contained in functions located in the 'head' of the page. JavaScript coding begins with a <script> tag. This can state the language and its version, but since the program is to be seen by students on the local server only, this information has been omitted from the tag illustrated here. An important feature of JavaScript programs is that they react to events such as a click of the user's mouse, keyboard entries, or page loading and unloading. This is illustrated by a function that is specified in the <body> tag when the page loads; in this case there are many other functions in the program above this one, so the <script> tag is not shown in the code (see Table 1(b)). The function simply removes the row of full stops in the last option of the 'select' list. It should be noted that the options are numbered from zero so the tenth one is number 9. Since the list box has already been placed on the page, removing the full stops does not resize it.

Functions responding to user input

Table 1(c) shows how the radio buttons respond to clicks by invoking the function loadm(number), where number indicates which button was clicked. The function uses this number to load a list with options (in this example these are the names of the modules). The code also introduces the 'if' statement, which is common to many programming languages and requires no explanation. It is necessary to determine which 'radio' button has been clicked; the variable is unambiguously called 'which'. Note that the protocol for the use of brackets is as follows: curly brackets,{ }, tie statements together so that they are used to indicate what should be performed in functions and also following the 'if' statement; square brackets, [ ], are used to indicate the index number of an array, in this case of options in the 'select' list; round brackets, ( ), are used when passing data to a function.

Opening new windows and writing to them

Table 1(d) shows how the effect of clicking one of the modules in the list is determined in part of the function whensel() that was given as the 'onchange' event handler (see Table 1(a)). In this function, there are a series of 'if' statements to determine which 'radio' button and modules have been selected. In the example shown, the file to be loaded is selected (fil) or in the third case, a new window is opened directly in response to a click. For most of the cases, the result of clicking the module name is that a new window opens and loads 'fil' into the left-hand frame of it (see Figure 2 and Table 1(e)).

This example illustrates several important features of JavaScript. Firstly, it is useful to open new windows, which reserve the whole screen for the information being presented to the students, rather than having space taken up by a menu bar, buttons and status bar. Only a scrollbar is required. Secondly, a very important feature of JavaScript is its ability to write to documents directly, which means that fewer HTML files are required. In this case, the result of clicking an option button is to define the file (fil), which is then opened into the left-hand frame of a new window. This frame takes up 25 per cent of the window, leaving the rest for notes on lecture material. The file loaded in the left-hand window has hypertext links that when clicked target the right-hand window. In this example, the learning resources given to the students include all the overheads used in the lectures (as 'gif' and 'jpg' files). These include animated flow charts and the example shown in Figure 2 illustrates how vitamin D is involved in calcium homeostasis by sequentially colouring

```
a)
For help with essay click start:<form name="f">
<input type="button" value="back" name="back" onclick="b()">
<input type="button" name="chbut" value="start" onclick="n()"><br>
<input type="button" value="Test" onclick="parent.location='teststar.htm'"></form>
b)
hint=new Array("n1-essay","vitc","vita","b12","megalo","zinc","iodine","thiamin","ribof","niacin","biotin")
count=-1
function n()
{if (count==-1) parent.frame1.document.f.chbut.value="Next";
if (count<10) {count++; parent.frame2.location=hint[count] + ".htm#hint";}}
function b()
{if (count>0) {count--;parent.frame2.location=hint[count] + ".htm#hint";}}
```

Table 2: Code relating to operation of Figure 2.

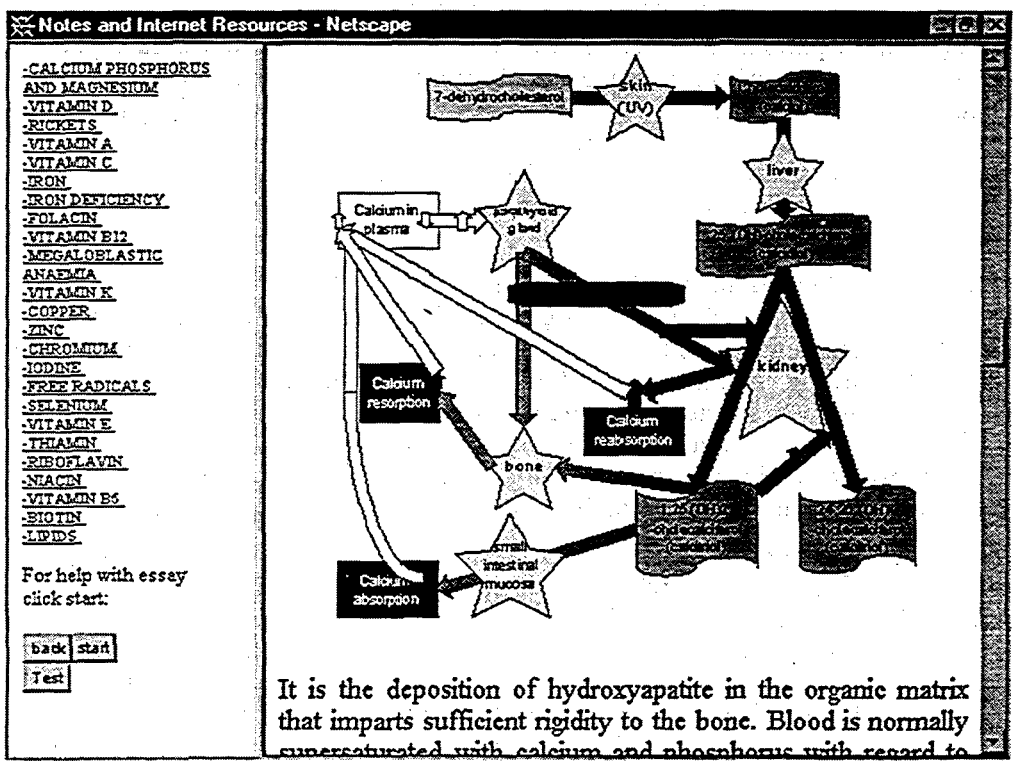

Figure 2: The presentation of lecture notes and overheads.

sections of the chart. The educational advantages of providing access to notes by students has been discussed by Barker (1995), and Mahalski (1995) has demonstrated that students make errors when copying from overheads during lectures and therefore should benefit from the availability of overheads on the computer for checking.

Use of arrays to define the response to user input

Table 2(a) shows how three buttons 'back', 'start' and 'test' are placed in the left-hand frame (see Figure 2). The responses of these buttons are defined by their 'onclick' properties. The first two buttons are used to help students with an essay; these buttons select the most appropriate sections of the notes in turn. Prior to providing this help to the students, it had been found that many students employed poor examples in their essay. The educational purpose of this essay is to develop the capacity to persuade the reader that a varied diet is beneficial and it is important that students use relevant examples. The code for the 'start' button includes a statement that changes the name on the button to 'next'; although the button is not movable, its caption can reflect the current state of the program. Table 2(b) shows that in the <script> tags in the 'head' of the document, there is an explicit definition of an array, in which the elements are stated; the array is called 'hint' and its contents are used to select the files for display in the right-hand window. The use of arrays is a very powerful tool in programming languages and JavaScript is no exception. The variable 'count' is defined earlier in the program and is used to remember which screen is currently being shown in the sequence; it is added to or subtracted from by 1 using 'count++' or 'count--', respectively.

Window hierarchy

Another feature of JavaScript is illustrated by the use of the word 'parent' in the script of Table 2(b). In the window (parent) there are two frames: the left-hand one has the list of parts of the notes and the buttons; the right-hand one shows the notes themselves. The frames are named frame1 and frame2. In order to refer to the file to be loaded into frame2, JavaScript uses the 'location' object, but the script is in frame1 and must refer to the other frame via the parent window, which contains both of the frames. JavaScript is an object-orientated language and there is a hierarchy of objects; in this case the window contains frames and these contain locations.

Use of loops to reduce script length

The button 'test' that was created in the previous example has a single 'onclick' instruction to change the location of the parent window. When clicked, therefore, this button will load a new file (teststar.htm); this removes all information from the window and sets it up again afresh. The test illustrates many useful features of JavaScript for self-assessment of students. Several different ideas have been used in a range of different screens, which use 'radio' buttons or check-boxes. The former type is used when only one answer is correct since only one 'radio' button in a group (with the same name) can be selected at once. The latter is used when there may be more than one correct answer, in this case true or false sentences. Figure 3 shows a simple example in which students are asked to define the axes and some points on a graph to see if they have understood it correctly. The code for the page in Table 3(a) again uses JavaScript's ability to write to the page, but there is now an additional feature that reduces the amount of code required and makes it easy to adapt this page to other graphs or pictures that you want the students to interpret. The page is divided up by using a table with three columns. There is a loop in the second column of the

```
a)
<form name="f">
<table cellspacing=0 border=1 cellpadding=7 width=590>
<tr><td width="35%" valign="center">
<img src = "ni-q-drv.gif">
</td>
<td width="15%" valign="center">
<font size=1>
<script>
for(c=0;c<5;c++)
{document.write('<input type="radio" name="l">');
document.write(lft[c]);
document.write('<br>');}
</script>
</td>
<td width="50%" valign="center">
<font size=2>
<script>
for(c=0;c<12;c++)
{document.write('<input type="radio" name="r" onclick="answer()">');
document.write(rtt[c]);
document.write('<br>');}
document.close()
firstcheck()
</script>
</td>
</tr>
</table>
</form>
b)
function putsent(numb){if(count==0)
{document.f.tx1.value="Free erythrocyte protoporphyrin is elevated by iron deficiency anaemia.";
document.f.tx2.value="Transferrin saturation is increased by iron deficiency anaemia.";
document.f.tx3.value="Ferritin is always decreased in individuals who have low iron stores.";
document.f.tx4.value="Iron deficiency is characterised by hyperchromic microcytic anaemia.";}
c)
function answer(){flag=false;
if(count==0){
if(document.f.ch1.checked==true && document.f.ch2.checked==false && document.f.ch3.checked==false
&& document.f.ch4.checked==false){flag=true;}}
```

Table 3: Code used in the interactive self-assessment programs.

table beginning with the instruction 'for', which starts with c=0 and ends with c=4, so it places five 'radio' buttons on the screen. The words to be used for each 'radio' button are stored in an array defined in the 'head' of the page. The program decides if the student has clicked the right answer using 'if' statements; if correct, the next 'radio' button in sequence is selected for the student to match with the correct answer from the list on the right. A

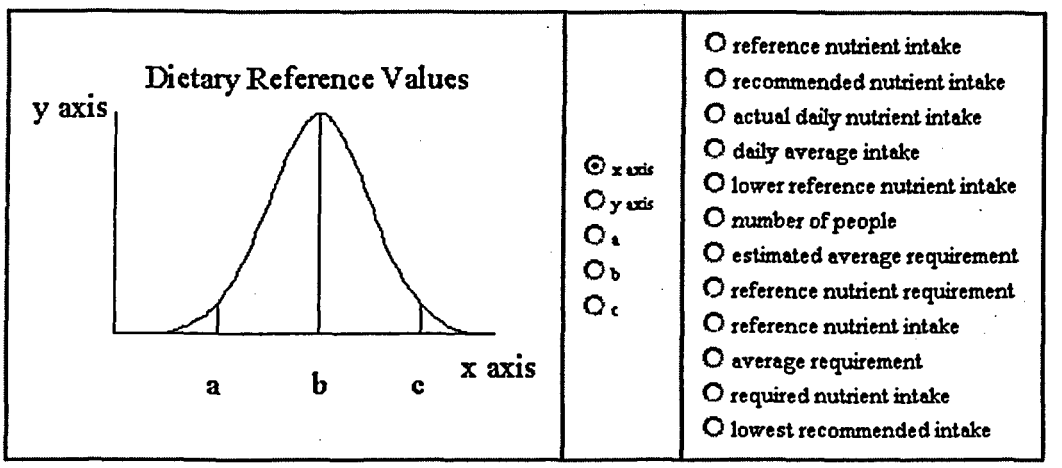

Figure 3: Interactive self-assessment screen.

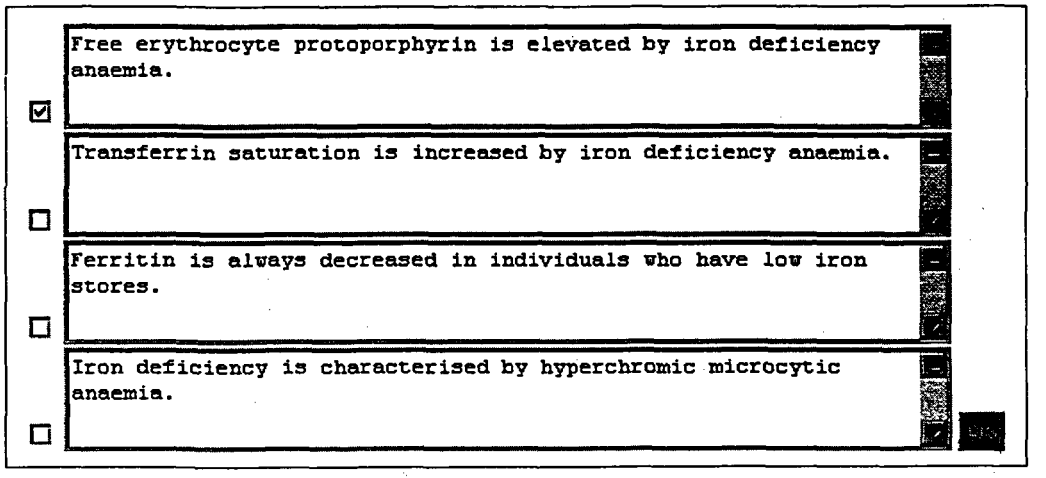

Figure 4: True or false sentences.

further example of this type of question screen, but omitting the figure, is one in which students have to match up signs of nutrient deficiency with a list of nutrients. A similar structure can provide a few sentences of text with many missing words; the words can be given on the right of the screen. Another more complex example asks students to match up two interacting nutrients, one in each column, with a statement about the reason for the interaction in a third column.

Figure 4 shows an example of a common type of objective question in self-assessment procedures. The true/false type of statement is useful because it can be made to test understanding as well as knowledge. The code example in Table 3(b) shows how the sentences are defined in a function that is invoked on loading the page and when the

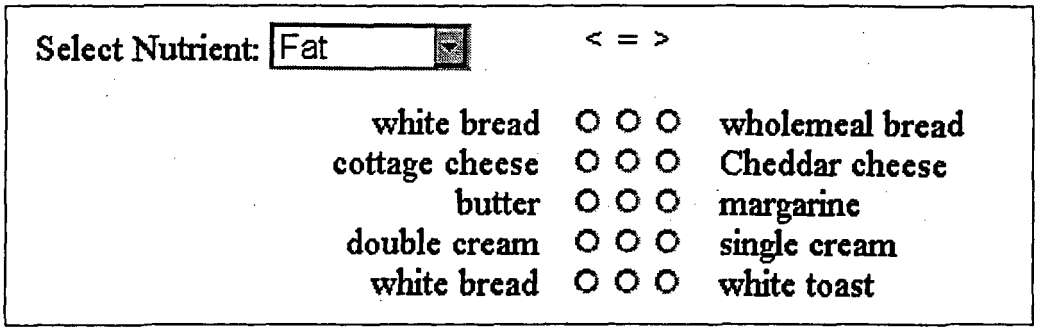

Figure 5: Comparison of nutrient contents of foods.

correct answer has been entered to allow a new set of sentences to be displayed. There can be many pages of these sentences in one HTML file and it is possible to have one or more true sentence per page so that students have to read all sentences carefully and not simply spot the first true sentence. Part of the code for the 'answer' function shown in Table 3(c) shows how this program determines whether the student can proceed or not; the output of this section 'flag' is true only if the correct sentences have been marked by the student. An objective test of this type is used as part of the assessment procedure for the module and this interactive test gives the students experience that helps them in the examination. It is important that students are able to determine from the output of the program that they have selected the correct answer, hence the program does not proceed until they have succeeded.

Figure 5 shows an example in which 'radio' buttons are combined with a 'select' list, this time with only one of the options visible and others available in a drop-down menu. The example requires students to state whether white bread has more, less or the same amount of fat compared with wholemeal bread. When students select the correct answer, an explanation of the reason is displayed to reinforce their knowledge. Students need to build a mind-map of the nutritional composition of foods and this is a useful method when combined with the use of a nutritional analysis program (Wise, 1998).

More advanced examples of JavaScript

Figure 6 shows a crossword puzzle developed using JavaScript that is used for revision; an example can be tried on the Internet (http://www.rgu.ac.uk/subj/fcs/diets.htm). This is being used in the tutorials to replace question and answer routines, which are considered threatening by some students. The clues can be designed to be cryptic and at times witty. This enhances the enjoyment of the students in completing the crossword puzzles whilst encouraging them to revise using other puzzles outside of timetabled hours. A version of the crossword puzzle written in 'Visual Basic' has been evaluated and was highly rated by students (Wise, 1998). The puzzle is developed by first thinking of relevant words and then entering these into the 'Crossword Construction Kit' (Insight Software Solutions, Bountiful, Utah (http://www.crosswordkit.com)). This then constructs an appropriate crossword. The results are then copied into a 'Visual Basic' program and clues added. The original 'Visual Basic' crossword program has been adapted to write the code directly so the process takes very little time and could be used by other staff who have no knowledge

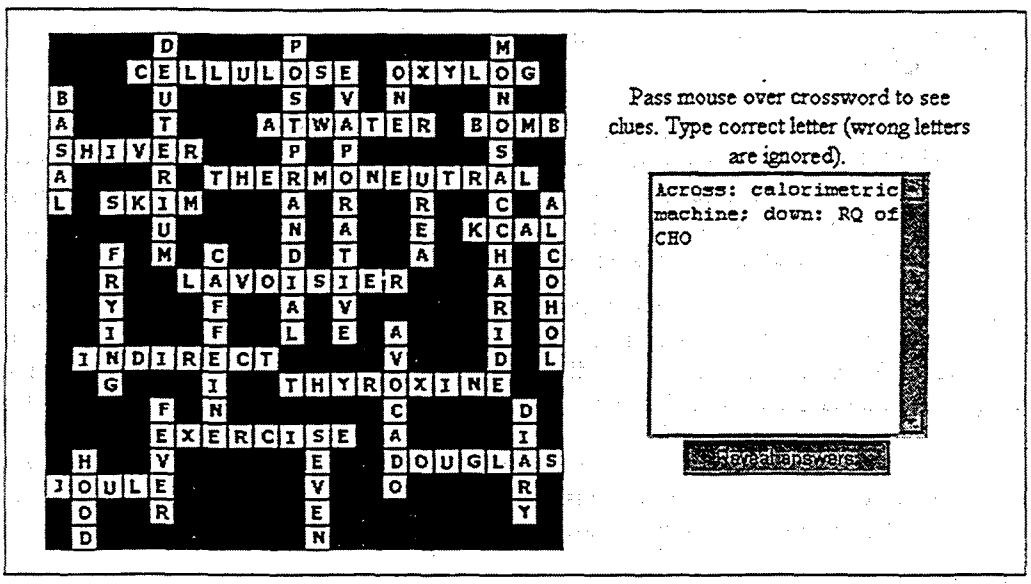

Figure 6: Crossword puzzle about energy.

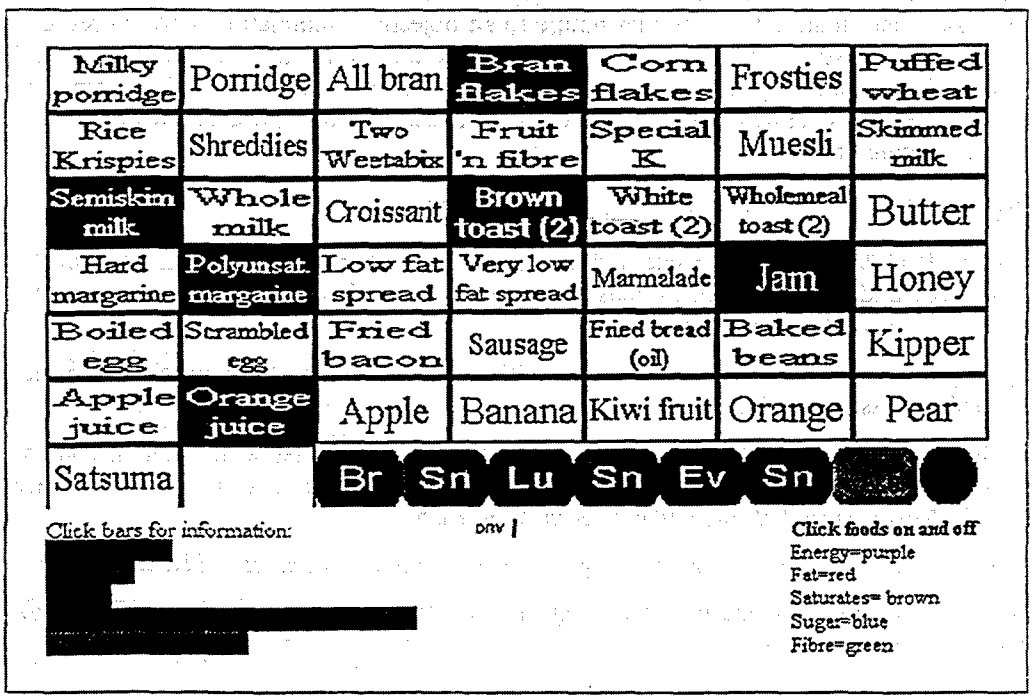

Figure 7: WebDiets: a simple program in which foods are selected by clicking them.

of JavaScript. Readers may wish to compare the JavaScript educational crossword with a commercial one (http://www.mirror.co.uk/crossword/index.html), which is written in Java and allows you to type any letter into the blank squares, even if it is wrong. The students currently have no access to this type of crossword program since their browsers are not Java-enabled, but if they did, this program would not be educationally satisfactory, because it does not indicate the correct answers. The JavaScript crossword also differs because clues are shown when the mouse hovers over the start of a word, it will only enter letters that are correct and it can reveal all the answers so that students learn from the activity.

Figure 7 shows the most advanced JavaScript program developed in this series. 'WebDiets' is a program that mimics a screen in 'WinDiets' (Univation Ltd, The Robert Gordon University, Aberdeen), a commercial dietary program used in teaching nutrition. In the program students select and deselect foods by clicking them, move from one meal to another and see the whole day's intake of energy and four dietary components in a bar chart, with a face that interprets the diet. Currently, our students cannot access 'WebDiets', since it requires a better browser, but it is available on the Internet for other students to try (http://www.rgu.ac.uk/subj/fcs/diets.htm).

Conclusions

JavaScript offers a simple solution to programming for interactive student support of learning. The type of material that can be provided includes lecture notes with overheads, self-assessment material of a similar nature to an objective examination, and fun revision aids such as crossword puzzles. No special configuration is required for the browser and potential problems arising from the existence of different dialects of JavaScript are relatively easy to deal with and are of no consequence if students have access only to one of the competing browsers.

References

Barker, P. (1995), 'Designing interactive learning support environments', in Percival, F., Land, R., and Edgar-Nevill, D. (eds.), *Computer Assisted and Open Access Education*, London: Kogan Page, pp. 46–51.

Flanagan, D. (1998), *JavaScript: The Definitive Guide*, 3rd edn., Sebastopol, California: O'Reilly & Associates, Inc.

Mahalski, P. A. (1995), 'What happens when students copy notes with different content and layout from an overhead screen? How often do they glance up and how accurate are they?', *British Journal of Educational Technology*, 26, 5–15.

Wise, A. (1986), 'Information technology in dietetic training', *Computer Education*, 53, 5.

Wise, A. (1998), 'Information technology in nutrition and dietetic education', *British Journal of Nutrition*, 79, 547–50.