Int. J. Engng Ed. Vol. 16, No. 1, pp. 50–55, 2000
Printed in Great Britain.

0949-149X/91 $3.00+0.00

Creating a Multiple-choice Self-marking Engine on the Internet*

T. W. NG
Faculty of Engineering EA-07-32, National University of Singapore, 9 Engineering Drive 1, Singapore 117576.
E-mail: engngtw@nus.edu.sg

Multiple-choice questions can be used to effectively reinforce concepts taught in lectures. This paper describes a multiple-choice self-marking engine which had been successfully developed using Javascript language. This tool can be easily incorporated into any web page written using the hypertext markup language (HTML) to provide students with a dynamic study aid over the Internet.

AUTHOR QUESTIONNAIRE

1. The paper discusses materials/software for a course in: Mechanics of Materials.
2. Students of the following departments are taught in this course: Mechanical and Manufacturing Engineering.
3. Level of the course (year): a first-year-level course in Mechanical and Manufacturing Engineering.
4. Mode of presentation: the tool described is available for students to access at any time over the Internet. They are encouraged to use it as often as they wish.
5. Is the material presented in a regular or elective course: Mechanics of Materials is an essential module which is covered over 39 hours of lectures, tutorials and laboratory sessions.
6. Class or hours required to cover the material: students will typically be required to complete one chapter of study before using the multiple choice tool to deepen their understanding.
7. Student homework or revision hours required for the materials: depending on the ability of the student, a range of 3 to 10 hours of revision would be needed before the tool would be useful.
8. Description of the novel aspects presented in your paper: this paper describes an Internet tool wherein educators can adapt to enable students to better comprehend a topic through multiple choice questions. The interactive nature of the tool makes study more interesting.
9. The standard text recommended in the course, in addition to author's notes: *Mechanics of Materials*, Beer and Johnston.

* Accepted 10 October 1999.

BACKGROUND

WITH THE INTERNET providing the means for students to access information remotely, the challenge for educators today is to develop tools that would engage students to deepen their understanding of a subject effectively. The presently ubiquitous hypertext markup language (HTML) was originally introduced to produce plain and static documents that could be viewed using any browser over the World-Wide-Web. Although many educators have since learned to mount their teaching material on web pages using this simple language, such pages, in honesty, can hardly hope to engage a student for more than one or two visits. The reason for this lies in the lack of interesting and interactive features on web pages based solely on HTML.

Creating interesting features on web pages, however, can appear to be daunting for an educator who has little background in programming. However, this does not have to be so with the Javascript [1–2] programming language. Javascript started its life as Livescript. When it was first introduced in January 1996 by Netscape, Livescript was designed to augment HTML pages. In the beginning, interest in Livescript was mild, due primarily to the frenzy surrounding a more robust Internet programming language called Java. When Netscape announced its decision to support Java, it did so by re-engineering Livescript, which was later renamed Javascript. Suddenly, interest in Javascript grew. The reasons were obvious. While Java required in-depth programming knowledge and a software development kit, Javascript needed none of these. As a scripting language, Javascript code can be easily prepared using any text editor and corrected on the fly without the need of compilation.

One project to produce more interactive educational web pages must surely be in the creation of multiple-choice self-marking engines. Although

multiple-choice questions are admittedly not ideal in eliciting creative answers from students, they are, nevertheless, invaluable in reinforcing important concepts taught in lectures. This is particularly pertinent for concepts that can be applied in a myraid of application instances. The main advantage of multiple choice testing is that the correction is objective. The major disadvantage is that writing good questions takes considerable skill and time. However, references are available that deal with the good practice of question design [3–5].

This paper aims to highlight a Javascript approach to create a multiple self-marking engine. There are, of course a myraid of ways, based on personal preferences, to produce such an engine. This paper does not attempt to discuss the various possible approaches. Rather the *modus operandi* is to state the desired features of an engine to be created and to describe how Javascript was used to engineer it.

DESCRIPTION OF THE MULTIPLE-CHOICE SELF-MARKING ENGINE

The engine described in this work was developed with the following features in mind:

1. The questions are to be randomly selected from a pool of questions prepared. If the student is presented with a question that he/she does not wish to attempt, it should be possible to skip that particular question.
2. The prepared questions are contained within a selection of HTML web pages.
3. A student responds to the question by a simple selection of choices, from A to D, using the computer's mouse.
4. The program compares the student's response with an attempted answer to the question and indicates whether it is right or wrong.
5. If the response given is wrong, the student should be able to reselect until he finally gets it right.
6. The engine does not grade the student's attempts as it is meant to be used as a tool by the student to gauge his/her understanding of the topics taught.

Clearly, a single HTML file would not suffice to achieve the features described. I have chosen to present the tool developed by giving the name of each document and showing a skeleton version of the text contained within. Following this, a description of the syntax is given to provide readers with a clearer understanding of what the instructions achieve. To facilitate description, the code rows are numbered with small letters along the left margin. Logically these numbers should be omitted if the code is to be used.

Filename: `quiz-header.html`
Contents:

```
1 ⟨HTML⟩
2 ⟨HEAD⟩
3 ⟨TITLE⟩This is the quiz header⟨/TITLE⟩
4 ⟨/HEAD⟩
5 ⟨FRAMESET ROWS = '30%,70%'⟩
6 ⟨FRAME SRC = 'processor.html'⟩
7 ⟨FRAME SRC = 'instruction.html' NAME =
  'results'⟩
8 ⟨/HTML⟩
```

This document creates a web page containing two frames for the quiz. The relative sizes of the frames can be controlled by the percentages input (line 5). The first frame (described by line 6) visually serves as the control console where the student can randomly generate a question and respond to that question. Within the HTML document supporting this frame (i.e. `processor.html`) is the Javascript syntax which generates the question and marks the response. The second frame (described by line 7), alternatively, serves as the frame where the quiz instruction and questions are placed. This frame contains standard HTML documents without Javascript. In other words, it functions as a dummy page in the interface. Since it is always good to include instructions for users, a HTML document with such content is called by default (e.g. `instruction.html`).

Filename: `instruction.html`
Contents:

```
1 ⟨HTML⟩
2 ⟨HEAD⟩
3 ⟨TITLE⟩This is the instruction page
  ⟨/TITLE⟩
4 ⟨/HEAD⟩
5 ⟨BODY⟩
6 ⟨P⟩Click on the Question Button to obtain a
  randomly generated question.
7 Click on selections A to D to choose your
  answer.
8 The program will automatically mark your
  response for you⟨/P⟩
9 ⟨/BODY⟩
10 ⟨/HTML⟩
```

This document contains the instruction for students (body of text given in lines 6 to 8). It is the default page which appears in the lower frame each time the tool is run.

Filename: `question-1.html`
Contents:

```
1 ⟨HTML⟩
2 ⟨HEAD⟩
3 ⟨TITLE⟩This is the quiz header⟨/TITLE⟩
4 ⟨/HEAD⟩
5 ⟨BODY⟩
6 ⟨P⟩This is Question Number 1⟨/P⟩
7 ⟨P⟩Selection A) Aa B) Bb C) Cc D) Dd⟨/P⟩
8 ⟨/BODY⟩
9 ⟨/HTML⟩
```

This is a sample HTML document containing the question. Each question is prepared in a separate document. It is possible to create many questions in different documents in this way. However,

the naming of the document is important. For instance, the second question should be named `question-2.html` and so forth. Note that it is possible to include accompanying pictures into these documents as well.

Filename: `processor.html`
Contents:

```
1 ⟨HTML⟩
2 ⟨HEAD⟩
3 ⟨SCRIPT LANGUAGE = 'Javascript'⟩

4 tieidx=0;

5 function ShowTie(){
6 tieidx=Math.round(Math.random() *
  (2-1))+1;
7 parent.results.location='question-'
  +tieidx+'.html';}

8 function CheckAnswer(sent){
9 answer = new Array()
10 A=1
11 B=2
12 C=3
13 D=4
14 answer[1]=B
15 answer[2]=D

16 if (tieidx==0){ window.alert('Select A
  Question First!'); }
17 else{
18     if (sent==answer[tieidx]){
19         window.alert('CORRECT
           ANSWER!... Very Good'); }
20     else{
21         window.alert('Sorry!       Wrong
           Answer... Try Again'); }
22     }
23 }

24 ⟨/SCRIPT⟩
25 ⟨/HEAD⟩

26 ⟨BODY⟩

27 ⟨FORM name = tiecheck⟩
28 ⟨Input   type='button'   value='Question'
   OnClick='ShowTie()'⟩
29 ⟨/FORM⟩

30 ⟨FORM name = tieform⟩
31 Select Your Answer
32 ⟨input type='radio' checked name='tiebox'
   onClick='CheckAnswer(1)'>A
33 ⟨input     type='radio'     name='tiebox'
   onClick='CheckAnswer(2)'>B
34 ⟨input     type='radio'     name='tiebox'
   onClick='CheckAnswer(3)'>C
35 ⟨input     type='radio'     name='tiebox'
   onClick='CheckAnswer(4)'>D
36 ⟨/FORM⟩

37 ⟨/BODY⟩
38 ⟨/HTML⟩
```

This is the HTML document that serves as the brain of the entire tool. There are two parts to this document. The portion of syntax contained within the ⟨SCRIPT⟩ tag (lines 3 to 24) is written in Java-

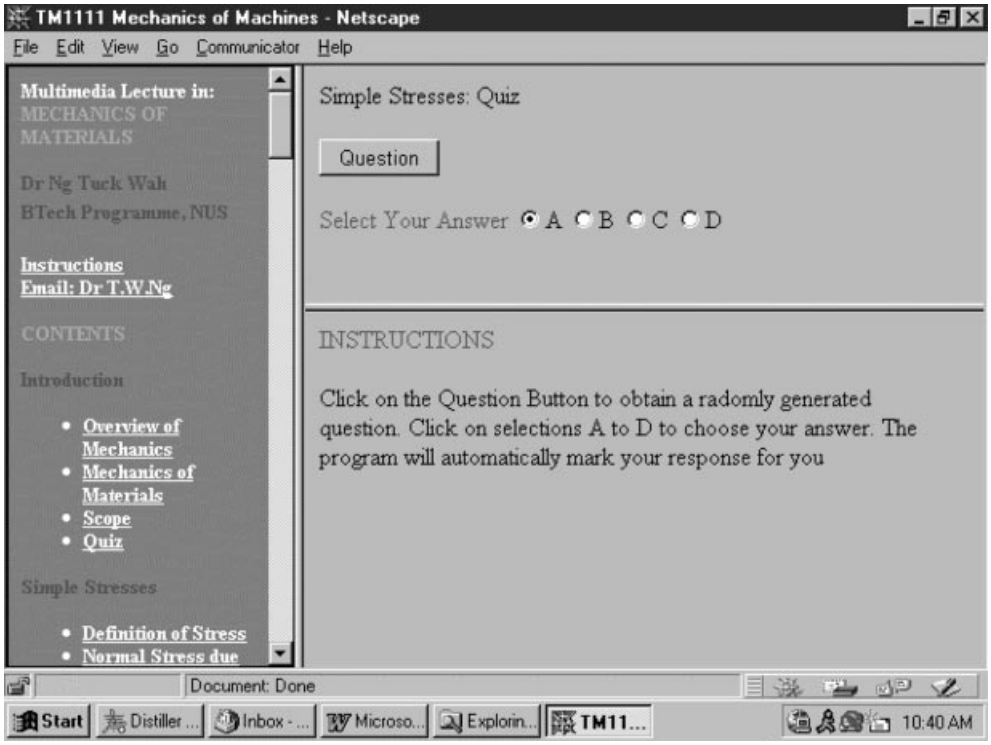

Fig. 1. The initial display of the multiple-choice self-marking engine as viewed using the internet browser.

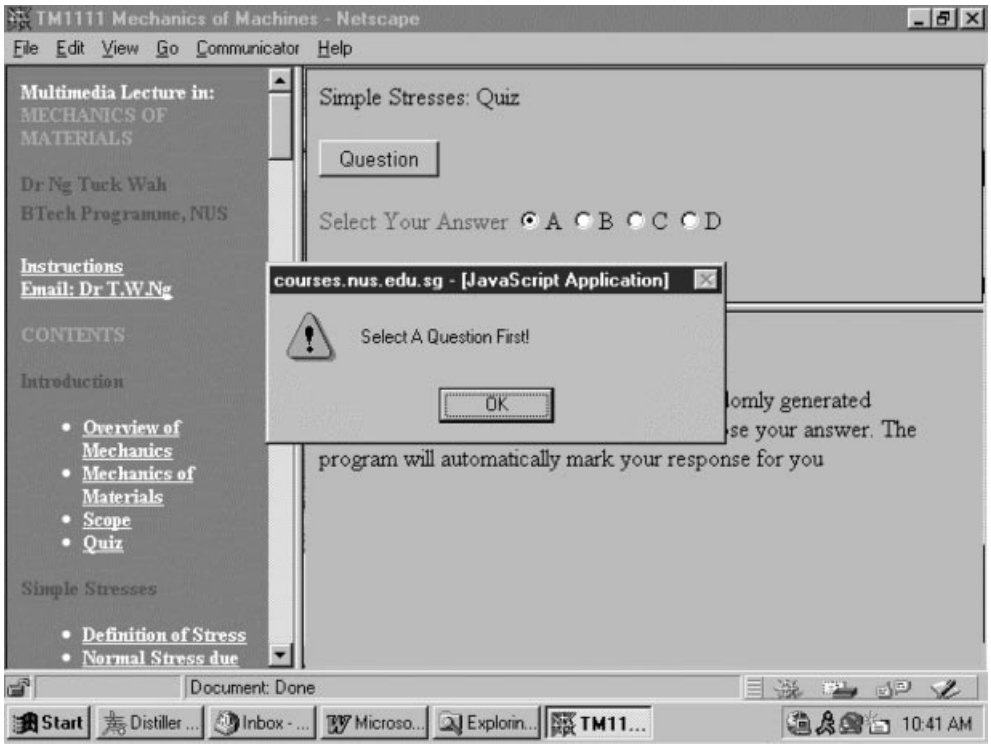

Fig. 2. The alert window appearing when an answer is given before a question is selected.

script and serves to generate a question and processes the student's selection. The other portion of syntax is written in standard HTML.

Let us consider the HTML code first. Notice that two forms are described. The first form, called tiecheck (line 27), appears as a conventional button. Once this button is clicked, the Javascript function `ShowTie(_)` is called. The second form, called tieform (line 30), contains radio buttons wherein a student can choose a response from A to D. Once a radio button is clicked, the function `CheckAnswer(_)` is called.

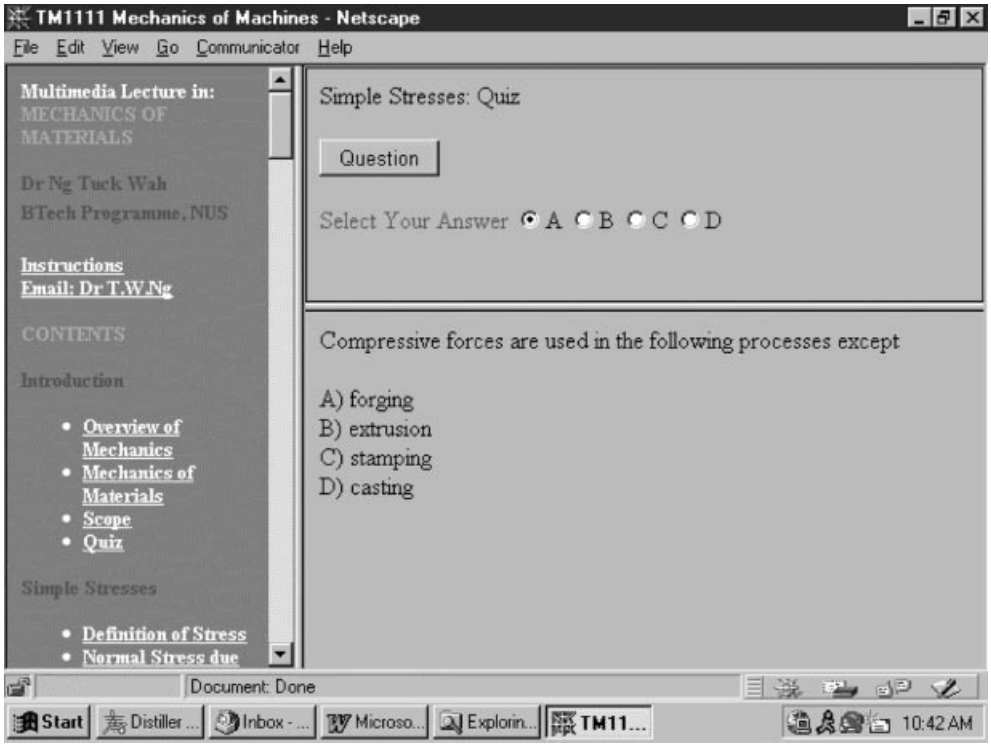

Fig. 3. Sample page when a question is selected.

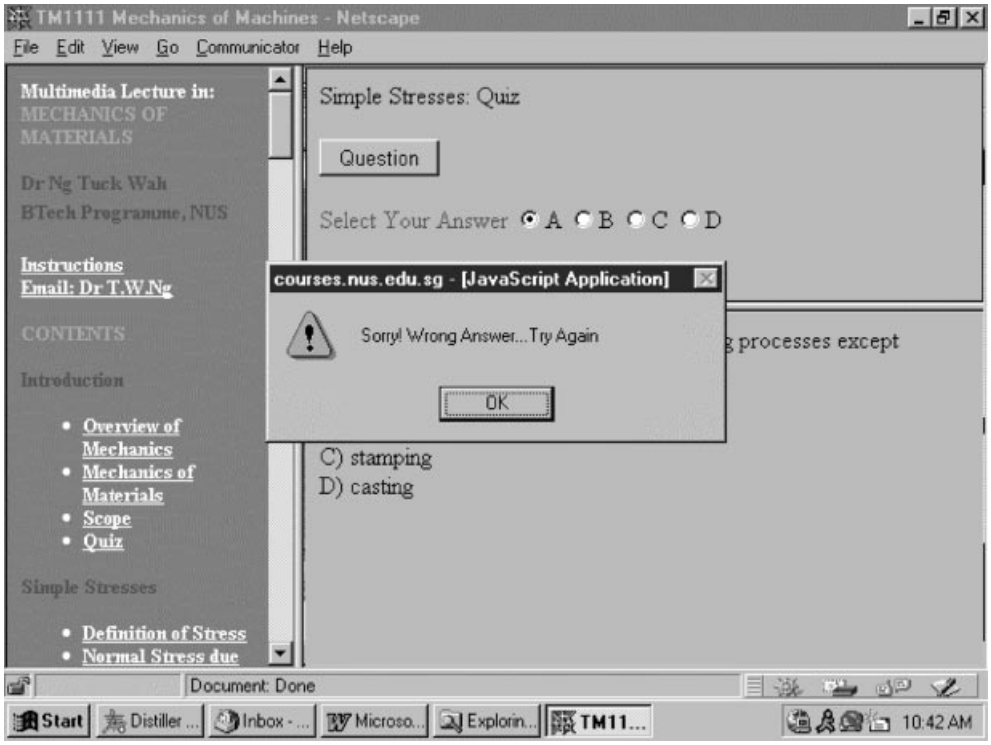

Fig. 4. The alert window appearing when the answer given is incorrect.

Let us now consider the Javascript portion of the document. Note that the variable `tieidx` is initially set to zero. Recall that clicking the conventional calls the function `ShowTie(_)`. A random number ranging from 1 to 2 is generated for tieidx. Of course, with X number of questions, the range of the random number can be changed to vary from 1 to X. A string is then generated to indicate the location of the HTML document containing the selected question. This is then placed in parent.results.location, which allows the question to be placed in the second frame described in document `quiz-header.html`.

We now consider the marking portion of the

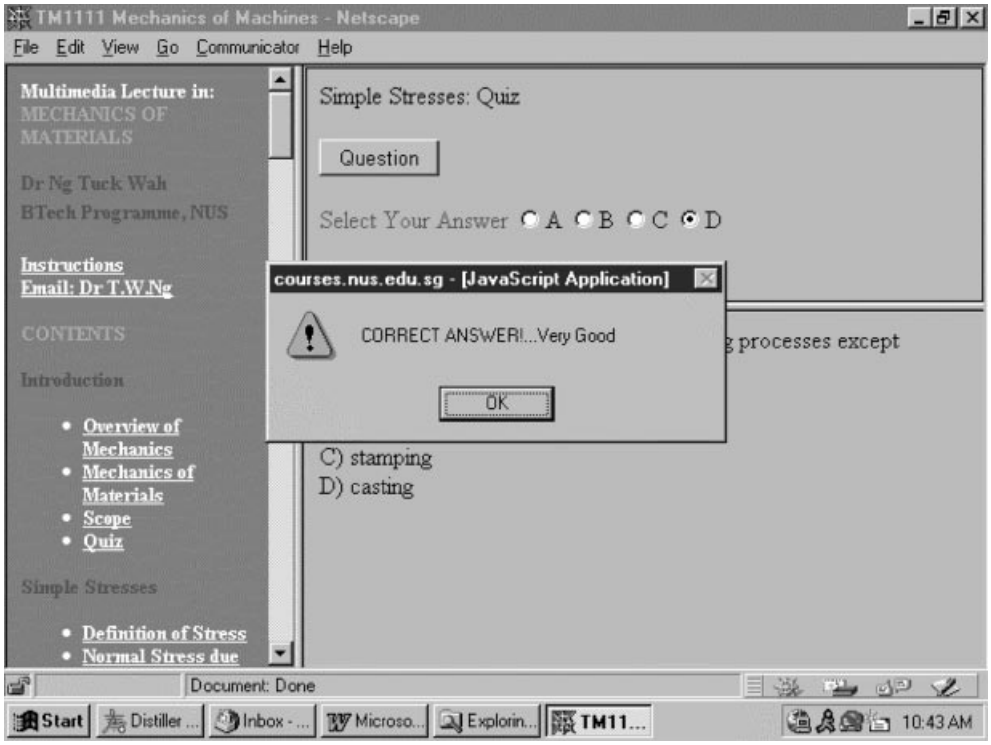

Fig. 5. The alert window appearing when the answer given is correct.

script. Recall that clicking any of the radio buttons calls the function `CheckAnswer(_)` (line 8). In this function, the variable `tieidx` identifies the question selected. If no question is selected, `tieidx` is zero. For this, a window will appear prompting the student to first select a question (line 16). If a question is selected, the marking can commence. All the answers to the question are placed in the array `answer[]`. The variable sent contains the response given by the student. If the contents of `answer[tieidx]` and sent are equal, it means that the student has answered correctly. A window will appear to make this known to the student (line 19). Conversely if the answer supplied is wrong, another window will appear to alert the student (line 21).

DEMONSTRATION

The multiple-choice self-marking tool developed was incorporated into a web page designed by the author to teach a first-year undergraduate course on the Mechanics of Materials. Sample pages illustrating the working of this tool are given in Figs 1 to 5. When the tool is first called, it appears as shown in Fig. 1. If a student attempts to give a response without selecting a question, a window appears prompting him to do so. This is shown in Fig. 2. Once a question is selected, it appears as shown in Fig. 3. Selecting the wrong response gives the window alert as shown in Fig. 4, whereas selecting the right answer gives the window alert as shown in Fig. 5.

CONCLUSIONS

This work describes and demonstrates a multiple-choice self-marking tool on the Internet developed using the Javascript language. The tool is simple and can be incorporated into any web page written using HTML. Such a tool can effectively engage a student to deepen his understanding of a subject by reinforcing the concepts learned in class.

REFERENCES

1. G. McComb, *Javascript Sourcebook*, John Wiley & Sons, New York (1996).
2. A. Weiss, *The Complete Idiot's Guide to Javascript*, Prentice Hall, Indianapolis (1997).
3. R. J. Leuba, Machine-scored testing, Part I: purposes, principles and practices, *Eng. Educ.*, **77** (1986) pp. 89–95.
4. R. J. Leuba, Machine-scored testing, Part II: creativity and item analysis, *Eng. Educ.*, **77** (1986) pp. 181–186 .
5. E. Boone and G. DeMay, Some practical aspects of multiple choice examinations, *Eur. J. Eng. Educ.*, **12** (1987) pp. 325–328 .

Dr T. W. Ng is currently Assistant Professor in the Faculty of Engineering, National University of Singapore. His research interests are in optical testing and educational development. Dr Ng is also editor for Optical Testing Digest an electronic periodical supported by SPIE (The International Society for Optical Engineering).