CSCU9B2 Practical 5: Cookies, Forms and JavaScript

Aims:

- **To extend your JavaScript programming.**
- **To explore the reading and writing of cookies with JavaScript.**
- **To use JavaScript to validate user entry in a form.**

Please register your practical attendance: Go to the GROUPS\CSCU9B2 folder in your Computer folder and double-click on the Register icon. Ask a demonstrator if you need help or something goes wrong.

This sheet contains one checkpoint (see end of sheet).

This week's practical is about cookies, forms and using JavaScript. We are *not* aiming to make you expert JavaScript programmers, but this will extend your knowledge of JavaScript and what it can be used for in web pages. It may take more than this practical slot to complete, so be prepared to work on it in your own time.

In addition to resources such as W3 Schools, there are some simple examples of JavaScript at:

http://www.cs.stir.ac.uk/courses/CSCU9B2/resources/JSexamples/

Cookies

Do you have any cookies set in your browser? How can you tell?

- In Internet Explorer, open **Tools: Internet Options**
- Click on the **General** tab, and then click **Settings** (inside Browsing History). This brings up another panel, Website Data Settings.
- Click the **View Files** button. This will probably bring up a huge set of files.

The Cookies can be identified because their names start with **cookie:** (if you use Details view, then click on the Name tab in the file viewer, this will put the files into alphabetical order).

In fact, you can make them easier to find by simply clicking the Delete Files button (beside the View Files button in Settings). This does not delete the cookies.

- Look at their names and their expiry dates. You can examine the cookies by opening them with TextPad or WordPad. What's in them? Not easy to tell: they are all coded. Nor is it easy to tell what the effect of altering them will be!

Later in this practical, we will set some cookies, and see what cookies are set, using JavaScript.

More Basic JavaScript

To extend your experience with JavaScript you will do some exercises that modify some of the example pages at: http://www.cs.stir.ac.uk/courses/CSCU9B2/resources/JSexamples/.

- Open this URL in Internet Explorer (IE) and try the different examples.
- Now copy folder **JSexamples** from **Wide\Groups\CSCU9B2** to your Web folder: this contains the code for these examples, which you will alter, as specified below.

Using the date object

- Go into your JSexamples folder and display the page **JSex3.html** (Example 3) using IE.
- Now also open it with TextPad so that you can edit it.

The JavaScript in this page is near the very end. Add JavaScript code to this script so that it displays "Happy Birthday" on today's date (or on your birthday) only. Do this by using the **today.getMonth()** and **today.getDate()** methods, together with a suitable *if statement*. Note that getMonth() actually returns a value between 0 and 11.

Here is an example *if statement* in Javascript:

```
if   (today.getDate()  ==  3) {
        document.write("<br> <b>It's the third today!</b>") ;
}
```

You use == to test for equality (remember to put the expression in brackets), and != to test for inequality. You can combine the conditions using &&, (logical AND), ||, (logical OR) and brackets. For example, to check for the second of December:

```
if  ((today.getDate() == 2 ) && (today.getMonth() == 11))
```

Look at the output of **today.getYear()**. What does it return on your browser? Originally, it returned just the last two digits of the year. Why (and when) do you think this was changed?

Using functions in JavaScript

Functions are a concept used in many different languages. In general, functions are passed some values (called *parameters*), operate on these values, and may return some new values (usually called *return values*). Common JavaScript functions are relatively simple.

- Open **JSex4.html** (Example 4) in IE and also in TextPad.

The functions are in a script in the header part of the HTML page (that is between the <head> and </head> near the start). Please put your functions in the header as well.

- Briefly examine the functions in Example 4, then scroll down the page to find where these functions are called: you will find that the calls are associated with buttons.
- Also note that there are other JavaScript scripts in the body of the page.

One of the main uses of JavaScript functions in web pages is to allow buttons to execute (groups of) JavaScript statements. Thus, in Example 4, pressing the first button calls the function **newtext()**, (a function which is not passed any parameters, hence the empty brackets) which changes the text in the adjacent text input form field. This function doesn't return any values either: it simply alters something on the page. Alter this part of the program (i.e. the function newtext()) so that:

- When the button is pressed, the text "DO NOT PRESS THIS BUTTON AGAIN!" is displayed on the button;
- When the button is pressed for a second time, and subsequently, the text "I told you not to do that!" is displayed on the button instead.

Hints: you will need to have some way of distinguishing whether the button is being pressed for the first time or not. One good way of doing this is to have a variable which is initialised to 0 when it is declared. The code:

```
<script>
var whichtime = 0 ;
</script>
```

will declare such a variable, but note that the variable needs to be declared before the function is encountered, not inside the function. Declaring it inside the function would mean that it was created each time the function was called, and destroyed each time the function was left.

You can the test this variable, and if it is 0 (whichtime == 0) the first text should be displayed, and the variable incremented. If it is non-zero (whichtime != 0) the second text is displayed.

As you will see from the rest of the example, functions can be used to implement a counter, counting how often a button has been pressed.

Advanced: Look at the code of the function **update()**. If you think your programming skills are up to it, alter the code so that instead of counting upwards, pressing the button causes the count to be decreased, starting from 10. When the value is 0, display the word "Ignition" instead of the number. When the button is pressed again, display the word "Lift-off".

Using JavaScript to set and read cookies

On the JavaScript examples web page, there are some examples which set and read cookies. There are a number of ways of setting cookies, but it cannot be done directly in HTML. Here, we use JavaScript to set and to read cookies.

Note that your browser must be set to permit cookies before this will work. Look in **Tools: Internet options**, and select the **Privacy** tab. Ensure that the privacy level is set to medium or lower. (You can always reset it later).

- Open your local copy of **JSwritecookies.html** in IE. This example writes and reads cookies and should create a cookie called **Chocolate**.
- Edit this HTML file in TextPad, and alter the name of the cookie it sets, and run it again. This time you should see two cookies, one called Chocolate, and the other named whatever you called it.

- Run the cookie reading page from the URL given above. This should show either no cookies, or just the original cookie, depending on whether you ran the write cookies page from there.
- Now open your local copy of the cookie reading page (**JSreadcookies.html**) in IE. This time you should see the two cookies.

What happens if the privacy level is set high?

Using JavaScript to validate a form

- Open your local copy of **JSex5.html** (Example 5) in IE and TextPad.

This page displays a simple text entry form. As it stands, it simply checks to see if there is any input at all. One can check the precise text of the input as well by simple comparison. However, it is useful to make the input value a string variable, because one can then use supplied methods to (e.g.) convert the text to lower case. We will try this:

- Inside the function **test(form1)**, declare a String variable, and initialise it from the input provided in the form e.g.:

  var str = new String(form1.text1.value) ;

When the function **test** that validates the form (i.e. the function called by onClick when the button is pressed) is called, this variable **str** will be created and initialised. It will be an object of type String.

Calling the String method: str.toLowerCase() will convert the text in the text field to lower case. Then you can use a test like:

(str.toLowerCase() == 'no')

to test for whether the string str contains no, No or NO (or indeed, nO).

- Use this approach to modify **test** to see if the text field contains *yes* or *no* (written in any mix of upper and lower case letters). Print out a suitable *alert* box (see the example for how to do this: it uses the function alert()) if the text is invalid (i.e. not *yes* or *no*).

CHECKPOINT [COOKIES]

Please show the lab demonstrator all your JavaScript code for these exercises, and demonstrate that they work. You may be asked for a short explanation of the code. The *advanced* part is not necessary for the checkpoint.