COMP519 Practical 10
JavaScript (5)

Introduction

- This worksheet contains further exercises that are intended to familiarise you with JavaScript Programming. While you work through the tasks below compare your results with those of your fellow students and ask for help and comments if required.
- You might proceed more quickly if you cut-and-paste code from this PDF file. Note that a cut-and-paste operation may introduce extra spaces into your code. It is important that those are removed and that your code exactly matches that shown in this worksheet.
- The exercises and instructions in this worksheet assume that you use the Department's Linux systems to experiment with JavaScript.
- To keep things simple, we will just use a text editor, a terminal, and a web browser. You can use whatever text editor and web browser you are most familiar or comfortable with.
- If you do not manage to get through all the exercises during this practical session, please complete them in your own time.

Exercises

1. From the last practical you should have a file `jsDemo09B.html` in your `$HOME/public_html/` directory containing the definition of a prototypical *employee* object. (If you have not completed the last practical, then it might be best if you skip this step and move directly to Exercise 2 in this worksheet.)

   a. We want to prompt the user to enter a new salary for one of our employees. Add the following code to the body section of `jsDemo09B.html` (after the already existing code):

   ```
   do {
     string  = prompt("Enter a new salary for "+
                      e[0].name+"?",e[0].getSalary())
     newSalary = parseInt(string)
   } while (isNaN(newSalary) || newSalary <= 0)
   e[0].setSalary(newSalary)
   alert("The new salary for "+e[0].name+" is "+
         e[0].getSalary())
   ```

   Save the file. Clear the output shown in the error console, then refresh the page in your web browser. Check that the code is working correctly. If it does you will first see a dialog box that prompts you to enter a new salary for 'Hal Smith' and then another dialog box will inform you what the salary of 'Hal Smith' has been changed to which should be the new salary that you have just entered.

   b. Obviously, that dialog would be much more useful if you could change the salary of an employee whom you specify by providing his/her name.

As a first step you would need an additional dialog box that prompts the user for the name of an employee. Then you would need to define a function that given a name finds which of the *employee* objects in the array `e` stores data for an employee with that name and returns that particular object; for comparison of names you could either use string equality or a regular expression search. Finally, you need to execute that function for the name that the user entered, and pass the returned object to the code in the previous step.

Implement that functionality and test it by entering various names that do or do not concur with the names for one of the existing employees and change their salaries.

2. Dialog boxes are a quick way of obtaining user input, but from an interface design point of view they are almost always the wrong choice for doing so. Forms are a much better way to build user interfaces.

   a. Create a file `jsForms.html` in your `public_html` directory with the following content:

```
<!DOCTYPE HTML>
<html lang="en-GB">
  <head>
    <title>JavaScript and Forms</title>
    <script>
      function FahrenheitToCelsius(temperature) {
        // Your definition here
      }
    </script>
  </head>
  <body>
    <h1>JavaScript and Forms</h1>
    <form name="form1" action="">
      Temperature in Fahrenheit:
      <input type="text" name="fahrenheit" id="df" size="10" value="0">
      <br>
      Temperature in Celsius:
      <input type="text" name="celsius"    id="dc" size="10" value=""
             onfocus="blur();">
      <br>
      <input type="button" name="Convert"
             onclick="document.form1.celsius.value =
                        FahrenheitToCelsius(parseFloat(
                        document.form1.fahrenheit.value)).toFixed(1);">
    </form>
    <div id="error"></div>
  </body>
</html>
```

   b. Make sure that the access rights `jsForms.html` are set correctly.

   c. Open `jsForms.html` in your web browser. You can enter a number into the first text field, but if you click on the 'Convert' button, then in the JavaScript console you will see a TypeError. This is because the function `FahrenheitToCelsius` does not return anything yet.

   d. Add code to `FahrenheitToCelsius` so that the function returns a value that is the equivalent in Celsius of the parameter `temperature` given in Fahrenheit. Save the file,

reload it in the web browser, enter a number into the first text field, then click on the 'Convert' button. In the second text field you should then see result of the conversion.

e. Enter a sequence of letters into the first text field instead of a number and click on the 'Convert' button. You should again see a TypeError in the JavaScript console. The problem obviously is that while we expect a number to be entered into the first text field, there is nothing that prevents the user from entering whatever they like.

f. Try to rectify this problem by replacing the first text field with a HTML5 form control that (only) allows to enter numbers.

g. Have you solved the problem? Enter the letter 'e' or 'E' into the first field. Is it accepted as input? If so, what is the result of converting it?

Make sure that you understand what is going on.

h. To rectify the problem we discovered in Exercise 2g, first create another function `processInput` that provides the same functionality as the code in the `onclick` attribute of the `Convert` button. Replace the code in the `onclick` attribute of the `Convert` button with a call of `processInput`, possibly with appropriate argument(s).

Now, within `processInput` check the user's input for correctness, say, using a regular expression. If the user's input is correct, proceed with the calculation of temperature in degrees Celsius and display result as before. If the user's input is incorrect, then display an error message in the `div` element with id `error` and put the focus back into the first field.

3. In the most recent lecture we discussed the implementation of a two-player board game using JavaScript. There were some tasks left to be done. In this exercise you should try to complete them.

a. Copy the file

`https://cgi.csc.liv.ac.uk/~ullrich/COMP519/examples/jsBoard.html`

to your `$HOME/public_html/` directory.

b. Define a function `checkWin` that checks whether one of the two players has managed to place three of his/her own pieces in a row, column, or diagonal on the board. If so, the function should return the number identifying that player (1 or 2) as result, otherwise it should return 0.

c. Define a function `showWin` that takes the number identifying a player as an argument and displays nicely styled message declaring that player to be the winner of the game, e.g. "Player 1 has won!". The function should make it impossible that the players can place further pieces on the board. This can be done either by removing the event handlers from all table cells or by establishing an end game state in which the play function does not make any more changes.

d. Within the play function add code at the appropriate point that calls the `checkWin` and `showWin` functions.

e. Within the play function add code at the appropriate point that checks whether there are free positions left on the board, calls an `endGame` function if there are not, and otherwise proceeds with the processing of the event.

f. Define a function `endGame` end the game with an appropriate message, e.g. "Game Over!". Again, the function should make it impossible that the players can place further pieces on the board. This can be done either by removing the event handlers from all table cells or by establishing an end game state in which the play function does not make any more changes.

4. Create a new file `jsRandom.html` with HTML and JavaScript code that provides the following functionality. Initially, the page shows the user a two-dimensional table with 3 columns and 3 rows where every cell of the table contains the number zero. Below the table should be a clickable HTML element with the label 'Calculate'.

   Whenever the user clicks on a cell, the number currently in the cell is replaced by a new random number between 1 and 9.

   If the user clicks on 'Calculate' a message box will be shown with the message 'The sum of all the numbers on the board is X' where X is the sum of all the numbers currently in the cells of the table.