Lab 12 – Unit Review

Aims:

- To practice how to create a data validation function for a HTML form using JavaScript;
- To review the JavaScript function and control structure;
- To prepare for the final exam;

Notes:

- This lab is not assessable.

Task 1: Form Styling and Validation

Description:

In this lab, you need to create a short message posting form that accepts two inputs, a text message and the student id that starts with "s". Then, style it using simple CSS. Finally, define a form data validation function using JavaScript.

Design:

Design starts with discussion and paper drawings. Ensure this process is completed before implementation.

Step 1: Form (HTML and CSS)

A mock up page had been design for the form as shown in Figure 1.

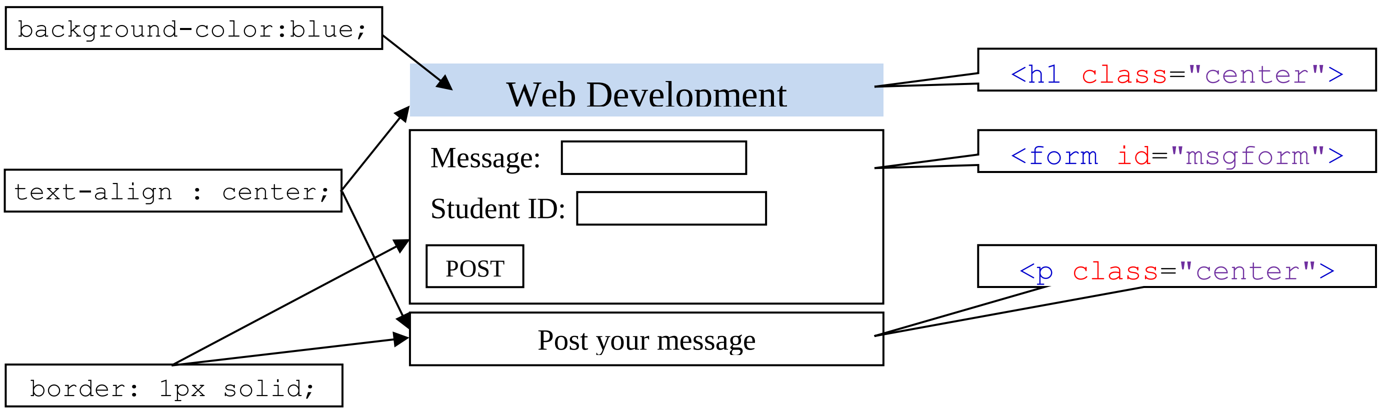

Figure 1. Form Mock Up

Step 2: Validation Rules

For this task, we need to validate all input fields.

The rules are

1) All inputs field must not be empty; and
2) Student ID must start with the letter 's'.

Implementation:

Implementation requires the creation of HTML, CSS and JavaScript files.

Step 3: Directory Set Up

3.1 Create a new folder 'lab12' under the unit folder on the mercury server `~/COS10005/www/htdocs`. This is the directory where all files will be uploaded.

Step 4: HTML Creation

4.1 Using NotePad++ (or Sublime Text for Mac users), open the text file `msgform.html`. Develop a web-based form as shown in the design – step 1 using HTML form and input elements. The form is to use the

POST method of data submission and the data will be processed by a server side script named `msgform.php`.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8" />
  <meta name="description" content="Web development" />
  <meta name="keywords"    content="Message Form" />
  <meta name="author"      content="put your name here" />
  <link href="msgform.css" rel="stylesheet" type="text/css" />
  <script src="validation.js"></script>
  <title>Web Development Message Form</title>
</head>
<body>
  <h1 class="center">Web Development</h1>
  ________________________________________________________________________________
  ________________________________________________________________________________
  ________________________________________________________________________________
  ________________________________________________________________________________
  ________________________________________________________________________________
  ________________________________________________________________________________
  ________________________________________________________________________________
  ________________________________________________________________________________
  ________________________________________________________________________________
  <p class="center">Post your message</p>
</body>
</html>
```

Step 5: CSS Creation

5.1 Open the text file `msgform.css`. Given the a web-based form as shown in the design – step 1 using 3 HTML content elements namely `<h1>`, `<form>` and `<p>`, write the appropriate CSS code.

____
____
____
____
____
____
____
____
____
____

Step 6: JavaScript Creation

6.1 Open the text file `validation.js`, write a client side JavaScript function called "`validateForm()`" that will validate the form.

- The validation of the form will be:
  - Ensure both the message and student id are not empty.
  - Ensure that the first character in the student id is letter "s".
- If there is a problem, show an alert box with a relevant message.
- Return **true** if the form data are OK, **false** otherwise.

```
/* write functions that defines the action for each event */
______________________________________________________________________________
______________________________________________________________________________
```

```
/* link HTML elements to corresponding event function */
function init () {
   /* link the variables to the HTML elements */
   var msgForm =  document.getElementById("msgform");

   /* assigns functions to corresponding events */
   msgForm.onsubmit = validateForm;
 }

/* execute the initialisation function once the window*/
window.onload = init;
```

Testing and Quality Assurance

Test your code for errors, this processes is also referred to as debugging. Use the Error Console provided by the Web Developer Firefox add-on. It can be accessed from the Web Developer toolbar: "Tools"->"Error Console". If there are errors, check if you missed any steps above.

[IMPORTANT] The Error Console provided by the Web Developer Firefox Add-on can help you identify JavaScript syntax errors. Please refer to the Error Console when your JavaScript does not work the way you want them to.

Step 7: Test and view web pages.

7.1 Using WinSCP, upload your files onto Mercury.
7.2 To view the pages through http, use any Web browser and type in the following address,

http://mercury.swin.edu.au/<your unit code>/***s***<*your Swinburne ID*>/<folder>/<filename>

Please refer to the following examples to identify the URLs of your web pages.

Folder on Mercury Web Server	URL
~/cos10005/www/htdocs/index.html	http://mercury.swin.edu.au/cos10005/s1234567/index.html
~/cos60002/www/htdocs/lab12/msgform.html	http://mercury.swin.edu.au/cos60002/s1234567/lab12/msgform.html

Note: You can copy the URLs in the table, but remember to replace the unit codes and student id in the above examples with yours to obtain the URLs of your web pages on Mercury.

[IMPORTANT] When the browser authorization request dialog pops up, use your SIMS username *and* password to confirm access, NOT your mercury username and password.

Step 8. HTML and CSS Validation

8.1 To validate the HTML file, use the Web Developer toolbar by clicking 'Tools'/ 'Validate HTML'. Alternatively, use the validator at http://validator.w3.org and for webpages pages on the server validate via 'URL'.

8.2 To validate the CSS file, use the Web Developer toolbar and by clicking 'Tools'/ 'Validate CSS', which will validate ALL CSS files linked to the html page at once.

Alternatively, validate CSS files using the 'File Upload' interface at http://jigsaw.w3.org/css-validator/ and upload all developed files.

Task 2: Condition formulation

1. Complete the client side JavaScript function `describeNumber()` that displays the range that a given number falls into .

```
function describeNumber(n) {
   var ans = "";

   if (_________________________){
     ans = "You entered a number between 1 and 9.")
   } else if (____________________________) {
     ans = "You entered a number between 10 and 19.")
   } else if (____________________________) {
     ans = "You entered a number between 20 and 29.")
   } else if (____________________________) {
     ans = "You entered a number between 30 and 39.")
   } else if (____________________________) {
     ans = "You entered a number less than 1 or greater than 39.")
   } else {
     ans = "You did not enter a number!")
   }
   return ans;
}

…
var n = prompt("Enter a number");
var ans = "You entered a number between " + describeNumber(n);
alert(ans);
```