Lab 09 – Document Object Model

Aims:

- To practice how to create a form data checking/validation function for a HTML form using JavaScript while maintaining clear separation of HTML, CSS and JS files.
- To review JavaScript functions and control structure
- To gain the skills and knowledge to complete Assignment 2

Task 1: Create Form Validation using JavaScript (10 marks)

Description:

Again we are going to start with a registration form, and then define and add a client-side form data validation function using JavaScript. In this lab, we will be using the JavaScript `alert` function to display the error message. As an extension, you may want to display the error message within the web page instead of using `alert`.

Remember to design the form interaction carefully before you start coding, i.e., when and what to do in response to what event.

Design:

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

Step 1: Form (HTML and CSS)

1.1 The design will be adapted from the form presented in Figure 1. For this lab, the error messages will be displayed using the JavaScript `alert` function, so no CSS solution will be needed. However, if you do decide to display error messages within the web page, you will need to design the interaction. You are encouraged to investigate and attempt the later approach.

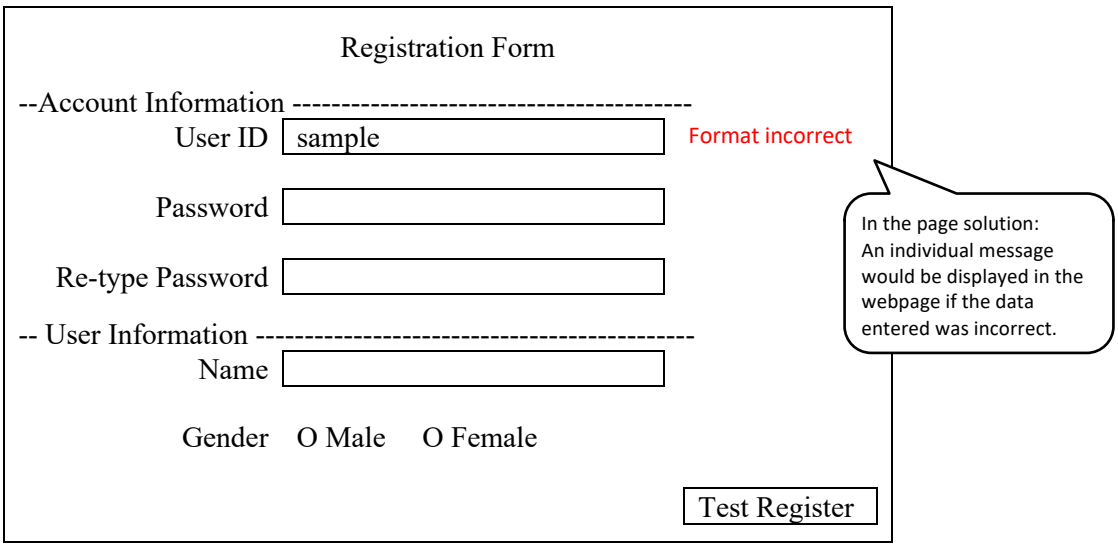


Figure 1. Form Mock Up

Step 2: (JavaScript)

2.1 Identify which input fields in the form should be evaluated and what rules should apply.

Answer: For this task, we need to evaluate all input fields. The rules are:
(1) All input fields must not be empty;
(2) User ID must contain at one '@' symbol to be a valid email address;
(3) Password and retype password must have the same value; and
(4) Name must be letters and spaces only.

Implementation:

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

Step 3: Directory Set Up

3.1 Create a new folder 'lab09' under the unit folder on the mercury server ~/COS10005/www/htdocs. This is the directory where all files will be uploaded.

Step 4:

4.1 Download `lab_09_files.zip` from the Canvas. Use the files in the zip file as templates for this lab.

Step 5: HTML Creation

5.1 Using NotePad++ (or SubLime Text for Mac users), open file `regform2.html`.
Review the HTML and complete the code.
For your convenience, the basic code and additional code is shown below. Make sure you understand the added code.

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="utf-8" />
   <meta name="description" content="Web development" />
   <meta name="keywords"    content="Registration Form" />
   <meta name="author"      content="put your name here" />
   (1)link to desktop CSS file
   (2)link to JavaScript data validation file
   <title>Web Development Registration Form</title>
</head>
<body>
   <form (3) enable JavaScript to uniquely identify the form method="post"
         action="http://mercury.swin.edu.au/it000000/cos10005/formtest.php">
      <fieldset>
         <legend>Account Information</legend>
         <div class="textinput">
            <label for="sid">User ID</label>
            <input id="sid" type="text" name="sid" />
         </div>
         <div class="textinput">
            <label for="pwd1">Password</label>
            <input id="pwd1" type="password" name="pwd1" />
         </div>
         <div class="textinput">
            <label for="pwd2">Retype Password</label>
            <input id="pwd2" type="password" name="pwd2" />
         </div>
      </fieldset>
      <fieldset>
         <legend>User Information</legend>
         <div class="textinput">
            <label for="uname">Name</label>
            <input id="uname" type="text" name="uname" />
         </div>
         <div class="radioinput">
            <fieldset>
               <legend>Gender</legend>
```

```
            <input id="genm" type="radio" name="gender" value="M" />
            <label for="genm">Male</label>
            <input id="genf" type="radio" name="gender" value="F" />
            <label for="genf">Female</label>
          </fieldset>
        </div>
      </fieldset>
      <div class="buttoninput">
        <input type="submit" value="Test Registration Form" />
      </div>
   </form>
</body>
</html>
```

Discussion

(1) link to desktop CSS file

Answer: `<link href="desktop.css" rel="stylesheet" type="text/css" />`

(2) link to JavaScript data validation file

Answer: `<script src="validation.js"></script>`

(3) add a form identifier, so we can easily reference the form object

Answer: `id="regform"`

Step 6: CSS Creation (for the form)

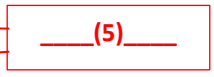
6.1 Open file `desktop.css`, review the CSS and the questions below.

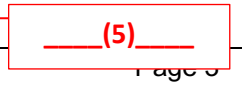
For your convenience, the code is shown below:

```
/* Style the form to a fixed width so the form looks the same even if the browser
   window is resize
*/
form {
   width            : 40em;
}
/* Style the labels and input by aligning all input fields
*/
label {
   float            : left;
   text-align       : right;
   width            : 10em;
   margin-right     : 1em;
}
input {
   width            : 50%;
}

/* Style text input to have spacing between input fields
*/
.textinput {
   margin           : 10px;
}

/* Style radio button input to a single line note that declarations are inherited
*/
.radioinput fieldset {
   border           : none; /*      (4)      */
}
.radioinput legend {
   float            : left;
   text-align       : right;
   width            : 9em;   /* Not 10em as it is inside a fieldset */
   margin-right     : 1em;
}
.radioinput input  {
```





```
   width           : 2em;  /* reset inherited value from 10em to 2em */
}
.radioinput label  {
   float           : none; /* clears the inherited float left */
   text-align      : left; /* reset inherited alignment to left */
}

/* Style button input note that declarations are inherited
*/
.buttoninput {
   text-align      : right;
}
.buttoninput input {
   width           : auto;
}
```

Discussion

(4) What happens if "border:none" is removed from the fieldset declaration?

Answer: The input for gender will be enclosed in a fieldset box, the word "Gender" – the legend - will be in the box, not in its default position along the top left of the box.

(5) Why is there a blank space and not a comma between these selectors?
However, in a previous lab we used a comma, for example

```
h1, h2, p {
  color : blue;
}
```

Answer: A blank space represents a **contextual** selector. This means that the style is to be applied **only** to HTML elements that are **descendants** of the class "`radioinput`". Note that when the border was set to none (as was asked in question 6) the fieldset for Account and User Information still retains its border.
"Comma" represents a **grouping** selector.

This means that the style rule is to be applied to all to these HTML elements. In the above grouping example, the `color:blue` will be applied to all h1, all h2 and all p, unless it is overridden by another later CSS declaration.

Step 7: JavaScript Creation (for the form data validation)

7.1 Open the text file `validation.js`, review the JavaScript and the questions below.
For your convenience, the basic code and additional code is shown below:

```
/* write functions that define the action for each event */
function validate() {
 var sid = document.getElementById("sid").value;
 var pwd1 = document.getElementById("pwd1").value;
 var pwd2 = document.getElementById("pwd2").value;
 var uname = document.getElementById("uname").value;
 /*     (6)     */
 var errMsg = "";                        /* stores the error message */
 var result = true;                      /* assumes no errors */
 var pattern = /^[a-zA-Z ]+$/;           /* regular expression for letters
                                            and spaces only */

 /* Rule 1, check if all required inputs have value */
 /*     (7)     */

 /* Rule 2, check if the user ID contains an @ symbol */
 /*     (8)     */

 /* Rule 3, check if password and retype password are the same */
 if (pwd1 != pwd2) {
    errMsg += "Passwords do not match.\n";
 }

 /* Rule 4, check if user name contains only letters and spaces */
```

```
    if (! uname.match (pattern)) {
       errMsg += "User name contains symbols.\n";
    }

    /* Display error message any error(s) is/are detected */
    if (errMsg != "") {
       alert(errMsg);
       result = false;
    }
    return result;
}

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

    /* assigns functions to corresponding events */
     regForm.onsubmit = validate;
}

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

(6) The property value has been used to access the text input, in the previous four line of code. What property needs to be used to access the radio input values?

Answer: The property is checked, thus to obtain the value stored in a radio input, the following JavaScript code needs to be added:

```
var genm = document.getElementById("genm").checked;
var genf = document.getElementById("genf").checked;
```

(7) How would you check if each input field is empty?

Answer: Add the following JavaScript Code:

```
if (sid == "") {
   errMsg += "User ID cannot be empty.\n";
}
if (pwd1 == "") {
   errMsg += "Password cannot be empty.\n";
}
if (pwd2 == "") {
   errMsg += "Retype password cannot be empty.\n";
}
if (uname == "") {
   errMsg += "User name cannot be empty.\n";
}
if ((genm == "")&&(genf == "")) {
   errMsg += "A gender must be selected.\n";
}
```

`\n` will create a 'new line' code in the message, that will be displayed in the `alert`

Note: You need to test all options in a radio input. In this case, both `genm` and `genf` must be tested.

(8) How would you check if an input field contains an @ symbol?

Answer: Add the following JavaScript code:

```
if (sid.indexOf('@') == 0 ) {
   errMsg += "User ID cannot start with an @ symbol.\n";
}
if (sid.indexOf('@') < 0 ) {
   errMsg += "User ID must contain an @ symbol.\n";
}
```

Note: This is simplistic way to check if the input is an email address. A better solution would be to use a regular expression.

An example of a regular expression is included in this lab, to check if the name input field only contains only letters and/or spaces. See the lines of code that declare and use a variable pattern:

```
var pattern = /^[a-zA-Z ]+$/;

if (! uname.match (pattern)) {
   errMsg += "User name contains symbols.\n";
}
```

Step 8: More Data Validation (Optional)

8.1 Add two more rules to Password:

1. It must be at least 8 digits.
2. It must contain at least one uppercase letter and one lowercase letter.

This step is optional but a good exercise for data validation using JavaScript. Use the JavaScript code provided at Step 7 as reference.

Step 9: HTML5 Extensions (Optional)

9.1 HTML5 provides additional HTML attributes that will do some of the form data checking on the client side, reducing the need for JavaScript.

`required="required"`

This attribute ensures that a value must be provided.
If we added 'required' attributes in the input elements, this would remove the need for the JavaScript provided for Rule 1 above

`type="email"`

This new input type ensures that the value provided must be in an email format.
If the browser does not support HTML5, the input will be treated as type="text"
This would remove the need for the JavaScript provided for Rule 2 above.

`pattern`

This attribute uses regular expressions to check a value entered. If we added the attribute `pattern ="^[a-zA-Z ]+$"` to the HTML input for the uname', this would remove the need for the JavaScript provided for Rule 4 above

Note: HTML5 offers only simple validation. Complicated validation would require JavaScript, e.g., Rule (3): Password and retype password must have the same value.

Testing and Quality Assurance:

Test your code for errors, this processes is also referred to as debugging. If there are errors, check if you missed any steps above.

Note: The Error Console provided by the Web Developer Firefox add-on is extremely effective in picking up syntax errors in your JavaScript code. It can be accessed from the Web Developer toolbar: "Tools"->"Error Console".

Step 10: Test and view web pages.

10.1 Using WinSCP (or Filezilla for Mac users), upload your files onto Mercury.

10.2 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/lab09/regform2.html	http://mercury.swin.edu.au/cos60002/s1234567/lab08/regform2.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.

Step 11. HTML and CSS Validation

To validate the HTML file, use the validator at http://validator.w3.org.

To validate the CSS file, use the interface at http://jigsaw.w3.org/css-validator/.