Lab 10 – jQuery

Aims

- To practice how to use a JavaScript framework such as jQuery to enhance user interaction.
- To review JavaScript functions and control structure.

Task 1: Create Form Validation using jQuery library (2 Marks)

Description: This lab is to demonstrate the use of jQuery.

Part 1: Based on a registration form, implement the JavaScript functions for client-side form data validation using jQuery.

Part 2: Create a section collapse effect for the Account Information and User Information sections of the registration form.

Part 3: Modify the validation function to use an HTML+CSS pop window (similar to Lab 8).

A simple preview of the web page to implement in this lab is presented in Figure 2 on page 6.

> Note: A gif file named `objective.gif` that presents what to achieve in this lab can be found in `lab_10_files.zip` which is available on Canvas.

Design:

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

Step 1: Form Creation and Presentation (HTML and CSS)

The design presented in Figure 1 will be used.

1.1Add a "[-]" symbol beside each input section for the user to click to collapse the section. Once a section is collapsed, it should display [+] to expand the input section.

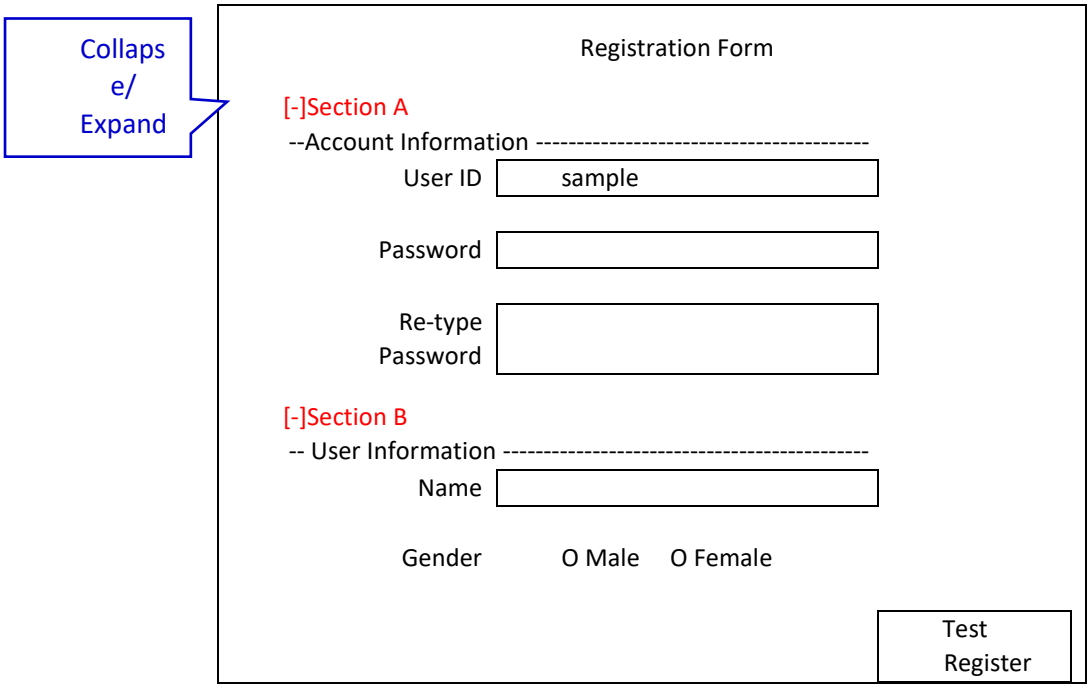

Figure 1. Form Mock Up

Step 2: JavaScript Implementation

2.1 Identify which what form data should be evaluated and what rules should apply.

Answer: We need to evaluate all input fields. The rules are:

Rule 1: All input fields must not be empty;
Rule 2: User ID must be a valid email address. Thus, it must contain a '@' symbol;
Rule 3: Password and Retype Password must have the same value; and
Rule 4: Name must be letters and spaces only.

Implementation

Implementation requires the creation of HTML, CSS and JavaScript files. In this lab, we will use the HTML and CSS files created in Lab 8. Those files are available in lab_10_files.zip on Canvas.

Step 3: Directory Set Up

3.1 Create a new folder 'lab10' 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 file regform2.html.

4.2 Review the HTML code and locate comments #1 - #6 and add missing HTML code as required.

For your convenience, the basic code and additional code is shown below:

```
<!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 modal CSS file for Task 3
(3)link to validation jQuery file
(4)link to validation JavaScript file
<title>Web Development Registration Form</title>
</head>
<body>
<form id="regform" method="post"
       action="http://mercury.swin.edu.au/it000000/cos10005/formtest.php">
  (5) add the [-]/[+] collapse/expand content
  <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>
  (6) add the [-]/[+] collapse/expand content
  <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="_____" rel="_____" type="_____" media="screen and (min-device-width:481px)" />`

2. link to modal CSS file

   Answer: `<link href="_____" rel="_____" type="_____" />`

3. link to jQuery library

   Answer: `<script src="_____"></script>`

4. link to validation window JavaScript file

   Answer: `<script src="_____"></script>`

5. write the [-]/[+] collapse/expand content for Account Information

   Answer: `<div><a class="collapse" href="#">[-]</a> <span>Section A</span></div>`

6. write the [-]/[+] collapse/expand content for User Information

   Answer: `<div><a class="collapse" href="#">[-]</a> <span>Section B</span></div>`

Discussion:

Numbers 5 and 6 above have the same answer apart from the section text. Attribute *class* is used instead of attribute *id*, as both "buttons" will allow similar user interaction which is to collapse the input section that follows it. The interaction will start with [-] as the form by default will be in expanded mode, and the user can click to collapse it. The [+] will be replaced dynamically with JavaScript using jQuery. [Extension: Try to make the two buttons look prettier using CSS. ☺]

Step 5: CSS Creation

5.1 Open files regform2_desktop.css and modal.css. Review the CSS code, no changes will be made to these files in this lab.

Step 6: Form Data Validation Using jQuery

6.1 Open file validation.js, convert the existing code to use jQuery as shown below.

Note: Replace '~~struck out~~' code with the **red** code shown below

```
/* function validate()will validate form data */
function validate() {

 var sid = document.getElementById("sid").value;          $("#sid").val();
 var pwd1 = document.getElementById("pwd1").value;        $("#pwd1").val();
 var pwd2 = document.getElementById("pwd2").value;        $("#pwd2").val();
 var uname = document.getElementById("uname").value;      $("#uname").val();
 var genm = document.getElementById("genm").checked;      $("#genm").prop("checked");
 var genf = document.getElementById("genf").checked;      $("#genf").prop("checked");

 var errMsg = "";              /* create variable to store 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 date are entered */
 if (sid == "") {              //check whether User ID is empty
errMsg += "User ID cannot be empty.\n";
 }
if (pwd1 == "") {   //check whether Password is empty

  errMsg += "Password cannot be empty.\n";
 }
 if (pwd2 == "") {  //check whether re-typed Password is empty
errMsg += "Retype password cannot be empty.\n";
 }
 if ((genm == "")&&(genf == "")) {                  (!genm)&&(!genf))
  errMsg += "A gender must be selected.\n";        //check whether gender is selected
 }

 /* Rule 2, check if the user ID contains an @ symbol */
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";
 }

 /* 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 () {
 /* assign the <form> element to variable regForm */  var
regForm = document.getElementById("regform");

 /* link function validate() to the onsubmit event of the form */  regForm.onsubmit
= validate;       $("#regform").submit(validate);
}

/* execute function init() once the window is loaded*/
window.onload = init;          $(document).ready(init);
```

Step 7: Collapse/Expand Effect Using jQuery

7.1 Open file validation.js, add the following function and code shown below.

Answer: Add the following function toggle () and event link code

---

```
/* write the function toogle() that collapse/expand a section*/
function toggle (){
   $(this).parent().next().slideToggle();      /* see explanation (7) below */

   if ($(this).html() == "[-]"){          /* Update the symbol on the "button" */
      $(this).html("[+]");
   } else {                               /* [-] <-> [+] */
      $(this).html("[-]");
   }
}

/* link HTML elements to corresponding event function */

function init() {
   $(".collapse").click(toggle);  //link function toogle() to appropriate events
   $("#regform").submit(validate);/*link function validate() to the submit event of the form */
}
```

This selects **all** elements in class "collapse"

7. $(this).parent().next().slideToggle();

   Answer: $(this) gets access to the element that triggered function toogle(), which is the [-]/[+] "button" created using an <a> element (see Step 4.2 Discussion #5 and #6). Then, function parent() is used to find the parent element of the [-]/[+] "button", i.e., the element that encloses the <a> element (see HTML presented in Step 4.2). After that, function next() is used to find the sibling element of the element, i.e., the <fieldset> element. Finally, jQuery function slideToogle() is called implement the collape/expand effect for the selected <fieldset> element.

Step 8: HTML+CSS Pop Window

8.1 Open the text file validation.js, modify variable errMsg and revise the code of function validate() as shown below.

8.2 This time, we first present each error as an <li> element. Later on, all those <li> elements will be placed in a <ul> element as an unordered list (see Step 8.3 below).

```
if (sid == "") {              //check whether User ID is empty
  errMsg += "<li>User ID cannot be empty.</li>";
}
if (pwd1 == "") {             //check whether Password is empty
  errMsg += "<li>Password cannot be empty.</li>";
}
if (pwd2 == "") {             //check whether re-typed Password is empty
  errMsg += "<li>Retype password cannot be empty.</li>";
```

```
}
if (uname == "") {                    //check whether User Name is empty
  errMsg += "<li>User name cannot be empty.</li>";
}
if ((!genm)&&(!genf)) {               //check whether gender is selected
  errMsg += "<li>A gender must be selected.</li>";
}

/* Rule 2, check if the user ID contains an @ symbol */
if (sid.indexOf('@') == 0 ) {
  errMsg += "<li>User ID cannot start with an @ symbol.</li>";
}
if (sid.indexOf('@') < 0 ) {
  errMsg += "<li>User ID must contain an @ symbol.</li>";
}

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

/* Rule 4, check if user name contains only letters and spaces */
if (! uname.match (pattern)) {
  errMsg += "<li>User name contains symbols.</li>";
}
```

8.3 Create an HTML element to cover the entire web page when the error message displayed (see Figure 2 below).

8.4 Place the error message into an unordered list and place the list in a section with a 'Close' button (see Figure 2 below).

8.5 Count the number of error, and store the value in variable numOfItems.

8.6 Add the error message HTML element immediately after the body element.

8.7 Make the screen overlay visible.

8.8 Adjust the window height and centre accordingly.

8.9 Make the pop up window visible.

8.10 Add a click event to disappear the screen overlay and pop up window.

```
/* Display error message if any error(s) is/are detected */
if (errMsg != "") {
  errMsg =   "<div id='scrnOverlay'></div>"                    //8.3
           + "<section id='errWin' class='window'><ul>"
           + errMsg                                            //8.4
           +  "</ul><a href='#' id='errBtn' class='button'>Close</a></section>";

  var numOfItems = ((errMsg.match(/<li>/g)).length) + 6;       //8.5

  $("body").after(errMsg);                                     //8.6
  $("#scrnOverlay").css('visibility', 'visible');              //8.7
  $("#errWin").css('height', numOfItems.toString() + 'em');    //8.8
  $("#errWin").css('margin-top', (numOfItems/-2).toString() + 'em'); //8.8
  $("#errWin").show();                                         //8.9
  $("#errBtn").click (function () {                            //8.10
                       $("#scrnOverlay").remove();
                       $("#errWin").remove();
                  } );

  result = false;
}
```

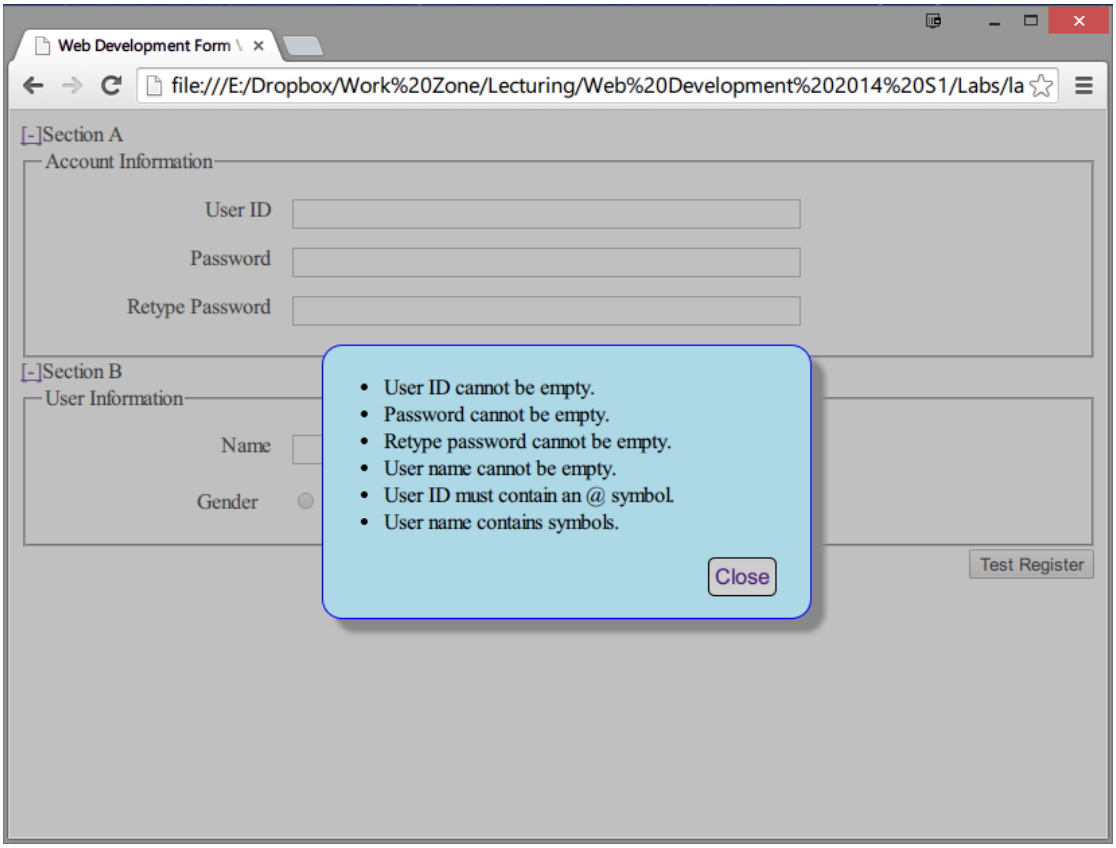

Figure 2. Error Message Displayed as a List in a Pop-Up Window

Testing and Quality Assurance

Test your code for errors, this process 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 syntax errors quickly. Please refer to the Error Console when your JavaScript does not the way you want them to.

Step 9: Test and view web pages.

9.1 Using WinSCP, upload your files onto Mercury.

9.2 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.

Step 10. HTML and CSS Validation

10.1 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'.

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.