Lab 8 – Javascript Part 2

Aims:

- To practice how to create interactivity for HTML pages using JavaScript and CSS, while maintaining clear separation of HTML, CSS and JS files.
- To review the JavaScript function and control structures
- To gain the skills and knowledge to complete Assignment 2

Note:

- Attempt both tasks prior to attending your scheduled lab.

Task 1: JavaScript for Interactivity (10 marks)

Description

We are going to create a 'pop up modal window' that displays a help list of the password rules to be displayed when the user clicks on a "Password Rule" 'button' located beside the password input field.

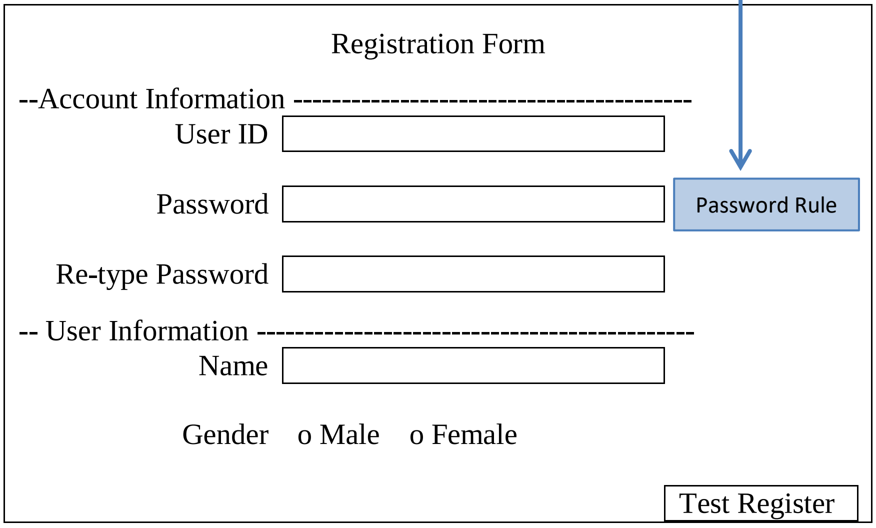
Remember to always design the form and interaction carefully before you start coding.

Design

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

Step 1: Form Design (HTML and CSS)

1.1 Using the form mock up presented below in Figrue 1, determine where the "Password Rule" button will appear. This button is not an HTML form button. It is an `<a>` element styled using CSS, not a HTML `<button>` element.



Registration Form

--Account Information --

User ID

Password [Password Rule]

Re-type Password

-- User Information --

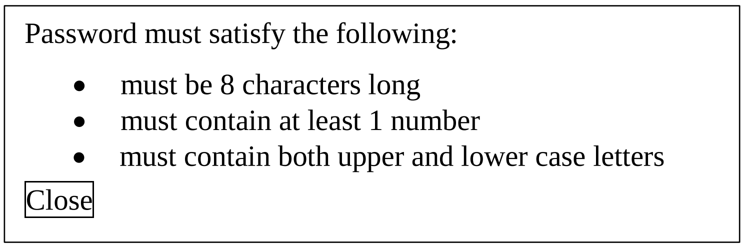
Name

Gender o Male o Female

[Test Register]

Figure 1. Example Mock Up

1.2 Draw a mock up of the pop-up modal window that will show the password rule list. Figure 2 presents an example mock up.



Password must satisfy the following:

- must be 8 characters long
- must contain at least 1 number
- must contain both upper and lower case letters

[Close]

Figure 2. Example Mock Up

1.3 When the modal window pops up, all input fields in the form window need to be disabled or made not selectable. What must be done to achieve this effect?
Answer: Display a full 'window layer' over the form window to prevent the user from being able to click on any input fields in the form. This 'window layer' will be a `<div>` element styled to cover the entire page. See Figure 3.

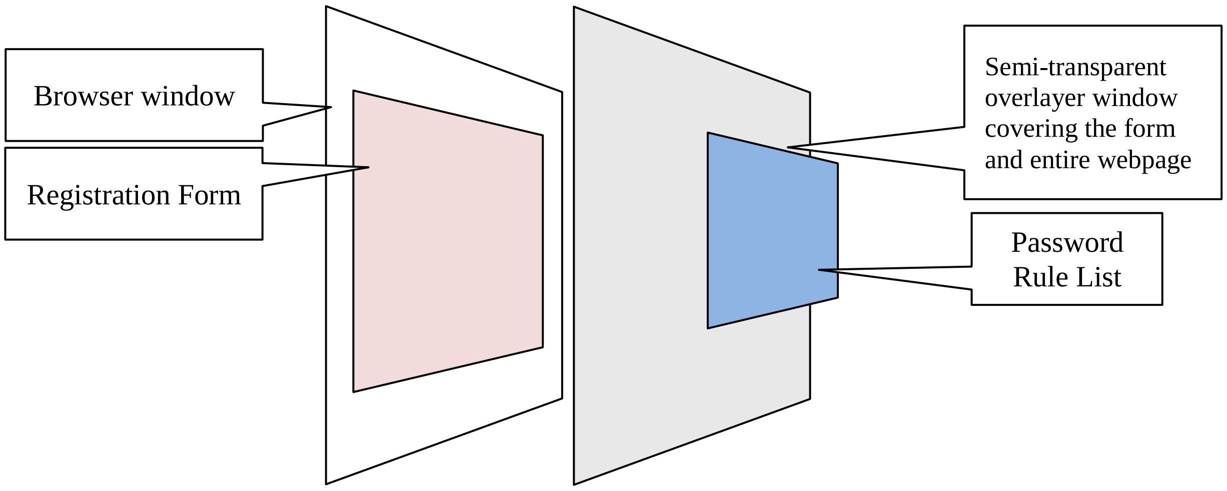

Browser window
Registration Form
Semi-transparent overlayer window covering the form and entire webpage
Password Rule List

Figure 3. A Semi-Transparent Window Layer Covering the Entire Web Page

Step 2: JavaScript

2.1 Identify which HTML element should trigger the interaction.
Answer: For this task, we need to create a Password Rule "button", i.e., a styled `<a>` element.

2.2 Identify what type of event or user interaction will result to a password rule list.
Answer: When the user clicks on the `<a>` element styled as a "button".

Implementation

Implementation requires the creation of HTML, CSS and JavaScript files.
In this lab, we will use the provided HTML and CSS files, i.e., `regform2.html`, `modal.css` and `desktop.css`.

Step 3: Directory Set Up

3.1 Create a new folder 'lab08' under the unit folder on the mercury server ~/cos10005/www/htdocs. This is the directory where all this labs files will be uploaded.
3.2 Download file `lab_08_files.zip` from Canvas. Extract the files from the zip file.

Step 4: HTML Creation

4.1 Using NotePad++ (or SubLime Text for Mac users), open the text file `regform2.html`.
Review the HTML and locate the 'comments' where we will add missing code.
For your convenience, the code below is provided as a template:

```
<!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 named desktop.css
   2) Link to modal window CSS file named modal.css
   3) Link to modal window JavaScript file named modal.js
   <title>Web Development Registration Form</title>
</head>
<body>
```

```
   4) create an overlay HTML <div> element

   <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" />
         5) write HTML code for the Password Rule Button
         6) write HTML code for Password Rule List
       </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="name">Name</label>
         <input id="name" type="text" name="name" />
       </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>
```

4.2 Complete and insert the following code into the right place in `regform2.html`.

1) Link to desktop CSS file

```
<link ___="________" rel="stylesheet" media="screen and (min-device-width:481px)" />
```

2) Link to modal window CSS file

```
<link ___="________" rel="stylesheet" />
```

3) Link to modal window JavaScript file

```
<script ___="________"></script>
```

4) Write the overlay HTML `<div>` element, identify it as “scrnOverlay”

```
<div ___="________"></div>
```

5) Write the HTML for the password rule button `<a>` element, identify it as “pwdHelpBtn”, put it in a class named “button”.

```
<a href="#" ___="________" ___="________">Password Rule</a>
```

Here, we used an `<a>` element is used and styled as a ‘button’. We did not use a `<p>` or a `<div>` element.

We did not use a `<span>` element as the mouse pointer would show as a text selector instead of a link selector. We did not use a `<input type="button">`, because this "button" is not part of a form.

6) Create the password rule list to be displayed in the modal window HTML.

```
<div id="pwdHelpWin" class="window">
<p>Password must satisfy the following:</p>
      <ul>
      <li>must be 8 characters long</li>
           <li>must contain at least 1 number</li>
           <li>must contain both upper and lower case letters</li>
      </ul>
      <a href="#" id="pwdHelpClose" class="button">Close</a>

</div>
```

A `<div>` element is used, as the content is to be displayed in a window. As this `<div>` element is to be displayed on another layer, it is not a requirement that it be placed after the password input field. We have chosen this placement to group the password list together with the "button". Again we have used an `<a>` element styled as a "button" to allow user clicks.

Step 5: CSS Creation (for the form)

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

For your convenience, the code below is provided as a template:

```
/*
   Style the form to a fixed width so the form looks the same even if the browser
   window is resize
*/
form {
   width             : 50em;
}
/* Style the labels and input by aligning all input fields */
label {
   float             : left;        /*___(7)___*/
   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;
}

.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

7) What happens if "`float:left;`" is removed from the label declaration?

Answer: The labels will be displayed by default as inline elements and 'width:10em' will not take effect. So be careful in CSS. When a width is specified, it does not necessarily mean that the element will have the specified width. It can be affected by other CSS declarations.

Step 6: CSS Creation (for the screen overlay, button and list window)

6.1 Open file `modal.css`, review the Discussion section to learn how the below CSS rules work.

For your convenience, the code below is provided as a template. See the discussion below the code.

```
/* ___(8)___ */
/* Use CSS to style the link as a button */
.button {
   border                  : 1px solid black;
   border-radius           : 6px;
   background-color        : lightgrey;
   padding                 : 0.25em;    /* Creates spacing around the text */
}

.button:hover {
   background-color        : darkgrey;
}


/* Use CSS to style the window   */
.window {                  /* apply style to HTML elements having window as class name */
   display              : none;         /*  ___(9)___   */
   z-index              : 2;            /*  ___(9)___   */
   position             : fixed;        /*  ___(9)___   */

   top                  : 50%;          /*  ___(10)___  */
   left                 : 50%;          /*  ___(10)___  */
   margin-top           : -5em;         /*  ___(10)___  */
   margin-left          : -11em;        /*  ___(10)___  */

   background-color     : blue;         /*  ___(11)___  */
   border-radius        : 5px;          /*  ___(11)___  */
   padding              : 5px;          /*  ___(11)___  */
   width                : 22em;         /*  ___(11)___  */
   height               : 10em;         /*  ___(11)___  */
}

/* set up a transparent pane on the screen, preventing the user from clicking on any
elements at layer 0 on the screen */

#scrnOverlay {
   visibility           : hidden;       /*  ___(12)___  */
   z-index              : 1;            /*  ___(12)___  */ /* use layer 1 to overlay
everything in layer 0 */

   position             : fixed;        /*  ___(13)___  */ /* Block out the whole screen */
   top                  : 0;            /*  ___(13)___  */
   bottom               : 0;            /*  ___(13)___  */
   left                   : 0;          /* ___(13)___ */
   right                  : 0;          /* ___(13)___ */

   background-color       : grey;       /* ___(14)___ */
   opacity                : 0.5;        /* ___(14)___ */
}
```

Discussion

8) What do the following CSS declarations do?

```
.button {
   border                : 1px solid black;
   border-radius         : 6px;
   background-color      : lightgrey;
   padding               : 0.25em;     /* Creates spacing around the text */
}

.button:hover {
   background-color      : darkgrey;
}
```

Answer: Apply a style of rounded rectangle around an HTML element, preferably an inline element. It will change the appearance of the button, when the mouse is placed over the styled element. We could later add other CSS such as gradient fill, and text-decoration none. ☺

9) What do the following CSS declarations do?

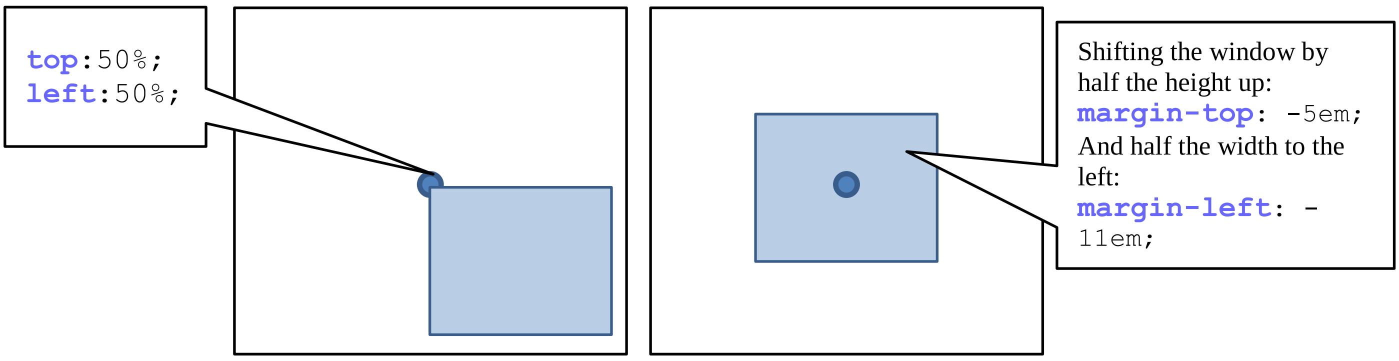
```
display              : none;
z-index              : 2;
position             : fixed;
```

Answer: Initially apply a style to **not** display the window class, unless the user clicks on the 'Password Rule' button. The window must then be displayed on top, (like 'bring to front' in Word or graphics editors, z-index 2 is the highest layer in our example.) And we want to display the window in a fixed location in the browser window not within the webpage. Remember, there is a difference between fixed and absolute position.

10) What do the following CSS declarations do?

```
top                  : 50%;
left                 : 50%;
margin-top           : -5em;
margin-left          : -11em;
```

Answer: Set the location where the window will appear. These four lines must be used together in order to place the window at the centre of the browser's window. Top and left must be 50%, while margin-top and margin-left is computed by dividing the window height and width by 2 and putting a negative sign.


top:50%;
left:50%;
Shifting the window by half the height up:
margin-top: -5em;
And half the width to the left:
margin-left: -11em;

11) What do the following CSS declarations do?

```
background-color       : lightblue;
border-radius          : 5px;
padding                : 5px;
```

```
        width                   : 22em;
        height                  : 10em;
```

Answer: Style the pop-up window's appearance. We could later add a border and shadow. ☺

12) What do the following CSS declarations do?

```
        visibility            : hidden;
        z-index               : 1;
```

Answer: Applies a style so the window is initially not visible, and it sits in a layer between the webpage and the pop up window.

13) What do the following CSS declarations do?

```
        position              : fixed;
        top                   : 0;
        bottom                : 0;
        left                  : 0;
        right                 : 0;
```

Answer: Apply style so the overlay covers the entire screen.

14) What do the following CSS declaration do?

```
        background-color      : grey;
        opacity               : 0.5;
```

Answer: Style how the overlay screen should appear, covering the entire screen but at the same time show the contents underneath using the opacity to control transparency. Try varying the number from 0 to 1.

Step 7: JavaScript Creation (for the modal window)

7.1 Open file `modal.js`, review the JavaScript and the questions below to learn how the below JavaScript code work.

Note: The colored rectangles indicate similar code for your reference.

```
/* function showPwdWin will show the Password Rule window */
function showPwdWin () {
  var pwdHelpWin = document.getElementById("pwdHelpWin"); /*get element "pwdHelpWin" */
  var scrnOverlay = ____________________________________;/*get element "scrnOverlay" */

  pwdHelpWin.style.display = "block";              /*display element pwdHelpWin */
  scrnOverlay.style.visibility = "visible";        /*hide element scrnOverlay */
}

/* function hidePwdWin will hide the Password Rule window */
function hidePwdWin () {
  var pwdHelpWin = ______________________________________; /*get element "pwdHelpWin" */
  var scrnOverlay = _____________________________________;  /*get element "scrnOverlay" */

  ____________________________________;     /* hide element pwdHelpWin by setting the CSS
                                               property display as "none" */

  ____________________________________;     /* display element scrnOverlay by setting the
                                               CSS property visibility as "hidden" */
}
/* link functions to appropriate events of corresponding HTML elements*/
function init () {
/* link the variables to the HTML elements */
  var pwdHelpBtn = ________________________________;  /* get element "pwdHelpBtn" */
  var pwdHelpClose = ______________________________;  /* get element "pwdHelpClose" */

  pwdHelpBtn.onclick = showPwdWin;     /* link function showPwdWin to the onclick event of
                                          button pwdHelpBtn */
```

```
    __________________________________;        /* links function hidePwdWin to the onclick event
                                               of button pwdHelpClose */
}

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

Testing and Quality Assurance

Step 8: JavaScript Debugging

8.1 Using the **Error Console** provided by the **Web Developer** Firefox extension or the **Firebug** Firefox extension to test your JavaScript code for errors. This process is also referred to as debugging.

8.2 Fix any errors you can find. Remember to refresh the web page every time you fix an error.

Step 9: Test and view web pages.

9.1 Using WinSCP (or FileZilla for Mac users), upload your files, including `regform2.html`, `desktop.css`, `modal.css` and `modal.js` 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.

Folder on Mercury Web Server	URL
~/cos10005/www/htdocs/index.html	http://mercury.swin.edu.au/cos10005/s1234567/index.html
~/cos60002/www/htdocs/lab08/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, NOT your mercury username and password.

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".

10.2 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.