

Lab 7 – Javascript: Part 1

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 JavaScript variables and expression
- To gain the skills and knowledge to complete Assignment 2

Task 1: Create Interactivity using JavaScript

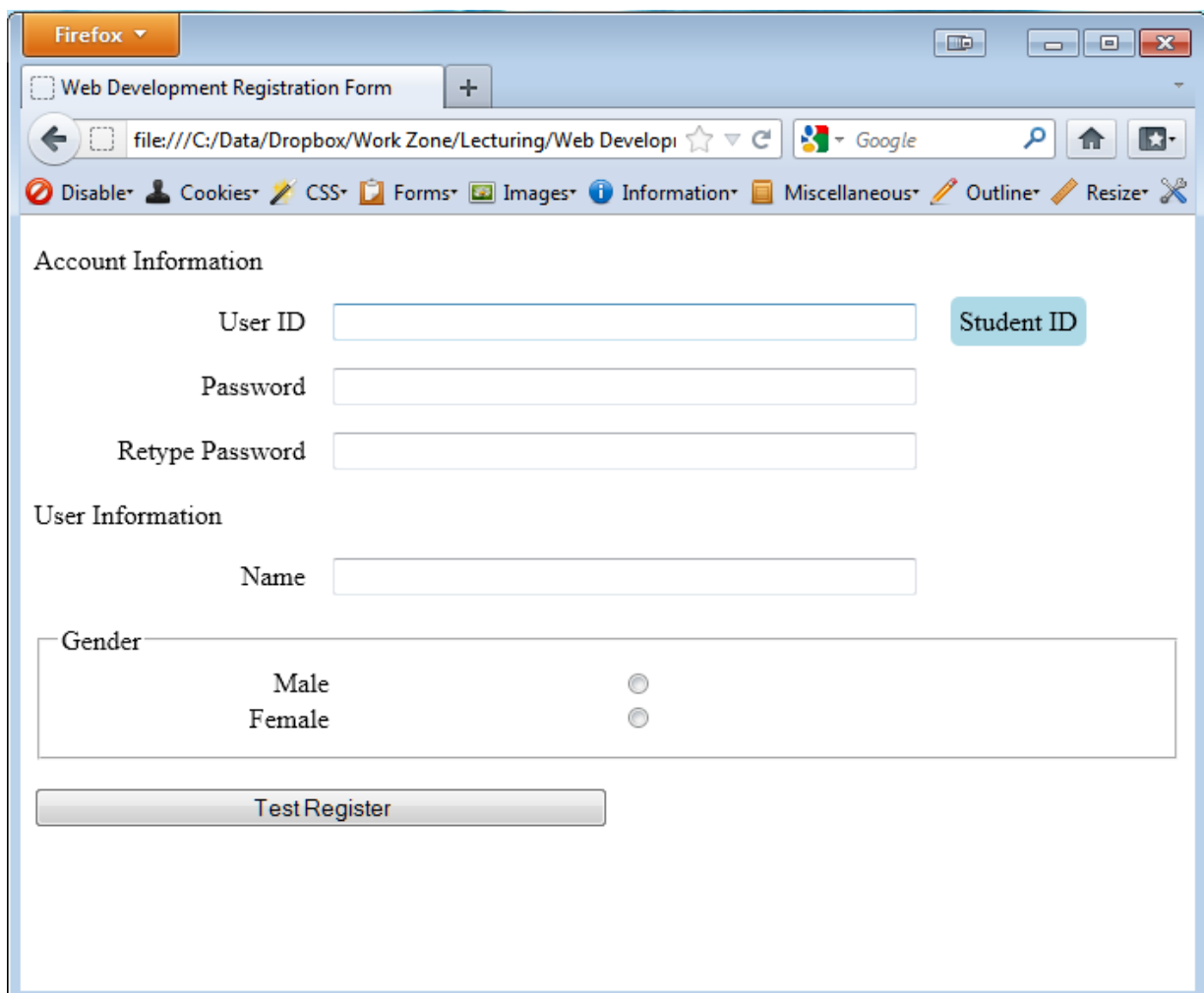
Description:

Dynamic behaviour can be added to the presentation of a Web page using **CSS** e.g. with **pseudo classes** like `a:hover` or `input:focus`, or through JavaScript. In this lab we will demonstrate the use of simple pseudo classes and JavaScript to enhance the user interaction, by displaying a 'tooltip' when the mouse pointer moves over an `<input>` element. Remember to always design the form carefully before you start coding.

Step 1. Design

Design starts with client discussions and white board and paper drawings. Always ensure this process is completed before implementation.

- 1.1 Draw a form mock up to illustrate the form, including the tooltip, which is to be presented using CSS. Figure 1 presents an example webpage.



The screenshot shows a Firefox browser window displaying a web page titled "Web Development Registration Form". The page contains two main sections: "Account Information" and "User Information".

Account Information:

- User ID:** A text input field.
- Password:** A text input field.
- Retype Password:** A text input field.
- Student ID:** A blue button.

User Information:

- Name:** A text input field.
- Gender:** A form with two radio buttons labeled "Male" and "Female".

At the bottom of the form is a "Test Register" button.

Figure 1. Example Mock-Up

Questions

1. Which HTML element should trigger the interaction?

Answer: For this task, we have identified the User ID text box.

2. What type of event or user interaction will trigger the display of the tooltip?

Answer: When mouse pointer moves over the user ID textbox the tooltip should appear.

When mouse pointer moves out of the user ID textbox the tooltip should disappear.

Step 2. Directory Set Up

2.1 Create a new folder 'lab07' under the unit folder on the Mercury server. Use this folder to store the files in this lab.

2.2 Download the zipped files from Canvas and use regform2.html as a template for this lab work.

Step 3. HTML Creation

3.1 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 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) Linkdesktop.css to desktop CSS file
  (2) Link to tooltip CSS file
  (3) Link to tooltip JavaScript file
  <title>Web Development Registration Form</title>
</head>
<body>
  <form method="post" action="http://mercury.swin.edu.au/it000000/cos10005/formtest.php">
    <p>Account Information</p>
    <p>
      <label for="sid" >User ID</label>
      <input type="text" name="sid" id="sid" />
      (4) write tooltip HTML
    </p>
    <p>
      <label for="pwd1">Password</label>
      <input type="password" name="pwd1" id="pwd1" />
    </p>
    <p>
      <label for="pwd2">Retype Password</label>
      <input type="password" name="pwd2" id="pwd2" />
    </p>
    <p>User Information</p>
    <p>
      <label for="username">Name</label>
      <input type="text" name="username" id="username" />
    </p>
    <fieldset>
      <legend>Gender</legend>
      <label for="genderM">Male</label>
      <input type="radio" name="gender" value="M" id="genderM" />
      <label for="genderF">Female</label>
      <input type="radio" name="gender" value="F" id="genderF"/>
    </fieldset>
    <p>
      <input type="submit" value="Test Register" />
    </p>
  </form>
</body>
</html>
```

- (1) Link desktop.css to regform2.html using `<link>`. Certain attributes are needed for `<link>` to work properly.
- (2) Link tooltip.css to regform2.html using `<link>`. Certain attributes are needed for `<link>` to work properly.
- (3) Link to tooltip JavaScript file to regform2.html using `<script></script>`. Certain attributes are needed for `<script>` to work properly.
- (4) Create the tooltip using ``:
`Student ID`

The span element is used, as the content is not a paragraph or another division.

Note that span is an inline element and thus will not occupy the full form width.

Note: You may later want to create a ‘tooltip’ for other input fields. Create other span elements, with the same class name for all tool tips, as this will be used by CSS to style the tooltip bubble. But the id must be unique for each element in order for the JavaScript to modify their visibility.

Step 4. CSS for the Form)

4.1 Using NotePad++ (or Sublime Text for Mac users), open the text file regform2_desktop.css and review the CSS and the questions below.

For your convenience, the basic code and additional 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 {
    _____ : _____; /* set the width to 40em */
}
/*
   Style the labels and input by aligning all input fields
*/
label {
    _____ : _____; /* float to left */
    _____ : _____; /* align text to right */
    _____ : _____; /* set width to 10em */
    _____ : _____; /* set right margin to 1em */
}
input {
    _____ : _____; /* set width to 50% */
}

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

/*
   Style radio button input to a single line note that declarations are inherited
*/
.radioinput fieldset {
    _____ : _____; /* remove borders */
}
.radioinput legend {
    _____ : _____; /* float to left */
    _____ : _____; /* align text to right */
    _____ : _____; /* set width to 9em */
}
.radioinput input {
    _____ : _____; /* set width to 2em */
}
```

```

}
.radioinput label {
    _____ : _____;          /* disable float */
    _____ : _____;          /* align text to left */
}

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

```

Questions

3. What happens if the form width is not set?

Answer: *It defaults to 100%, and generates a fluid layout.*

Modify the CSS and test the effect by resizing the browser.

4. Why is it preferable to use a class name instead of id for the input field, such as text, radio and submit button?

Answer: *This is to achieve a uniform look and feel for similar input types.*

5. Add the following additional **pseudo class** code to the end of the CSS:

```

input[type=text]:focus , input[type=password]:focus
{
    background-color:yellow
}

```

This will ‘highlight’ the input elements of type=text and type=password when they are given focus by the user.

Step 5. CSS for the Tooltip

5.1 Using NotePad++, open the text file tooltip.css, complete the CSS and review the questions below.

5.2 Complete the CSS code below based on the comments provided. It is used to style the CSS tooltip, i.e., the `` element created in Step 3.1

```

.tooltip {
    display          : _____;          /* apply style to HTML elements having a tooltip as
    position         : relative;            class name */
    _____      : blue;                /* tool tips are initially not visible */
    _____      : 5px;                 /* place tooltip according to the order in the HTML
    text-align       : _____;          */
    padding         : _____;          /* uses a blue background */
    left            : 1em;                 /* create a rounded corner */
    _____      : _____;          /* center text inside the tooltip */
    _____      : _____;          /* add a 5px spacing around the text */
    _____      : 1em;                 /* position the tooltip at 1em from the left */
}

.tooltip:after {
    position         : _____;          /* uses a pseudo class to apply style to the text
    top              : 10px;               that is appended to the tooltip */
    border-style     : _____;          /* set an absolute position be specified (default
    height           : _____;          is after) */
    width           : _____;          /* specify the absolute top position */
    _____      : _____;          /* specify solid border style*/
    _____      : _____;          /* set height to 0px */
    _____      : _____;          /* set width to 0px */
}

```

Questions

6. What do the following CSS declarations do?

```

border-color      : transparent blue transparent transparent;
border-style      : solid;
border-width      : 8px;

```

```
height      : 0;
width       : 0;
```

Answer: Displays the right border. If combined with a thick border (11) and empty content (12), creates an effect of a call out (triangle) pointing to the left.

Step 6. JavaScript Creation (for the Tooltip)

6.1 Using NotePad++, open the text file tooltip.js, create and complete function showTip () .

```
/* write functions that defines the action for each event */
function showTip() {
    _____; /* A variable named sidTip is defined */
    sidTip = _____; /* Get access to the element
                        with id "sidTip" */
    _____ = _____; /* The CSS property "display" of
                        element "sidTip" is set to
                        "inline" */
}
```

6.2 Create function hideTip(), using showTip() in step 6.1 as an example.

```
function hideTip() {
    _____; /* create a variable named sidTip */
    _____; /* get access to the element by
                its id "sidTip" */
    _____; /* hide element "sidTip" by
                setting the CSS property "display"
                to "none" */
}
```

6.3 Create function init(). In function init(), we link JavaScript functions to the appropriate events of corresponding HTML elements.

```
function init() {
    _____; /* create a variable named sid */
    _____; /* get access to the HTML element
                by its id "sid" and link it to sid */
    _____; /* link function showTip to the
                onmouseover event of sid */
    _____; /* link function hideTip to the
                onmouseout event of sid */
}

_____; /* link function init to the onload event of
the window so that function init will be
called when the page is loaded */
```

Questions

7. What is the general programming pattern that we used when writing a JavaScript code for HTML?

Answer: The code will be divided into three sections as follows.

Write functions that defines the action for each event:

```
function yourEventActionName () {  
    ...your JavaScript code goes here...  
}
```

Link HTML elements to corresponding event function:

```
function init () {  
  
    ...list and link all HTML elements to a variable...  
  
    ...link your event function to HTML elements...  
}
```

Execute the initialisation function once the window is loaded:

```
window.onload = init;
```

8. Have you considered issues of accessibility and usability for the enhancement? Why?

Answer: *No, because if the user uses mainly the keyboard, the tool tip will not appear even if the cursor is on the input field. It still requires the user to position the mouse pointer over the input field for the tool tip to appear. This can be resolved by linking additional event(s) to the show and hide tool tip functions. (Note this is a simplified solution.) Analyse the modified code below, is it the perfect solution? It depends on your usability criteria. If you move the mouse pointer in and out of the input field when the cursor is still in focus, you will see that the tool tip disappears.*

```
function init() {  
    /* link the variables to the HTML elements */  
    var sid = document.getElementById("sid");  
  
    /* links functions to corresponding events */  
    sid.onmouseover = showTip; /* for mouse */  
    sid.onmouseout = hideTip;  
    sid.onfocus = showTip;    /* for cursor on input field */  
    sid.onblur = hideTip;     /* for cursor moving out */  
}
```

Step 7. Tooltip for Password (Optional)

- 7.1 Create a tooltip bubble for the **password text box** on `regform2.html` by repeating Step 3 through Step 6. The tooltip bubble for the password text box provides information “6-character password”.

Step 8. Testing and Debugging

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

Step 9. Viewing and Testing Web Pages.

9.1 Using WinSCP (or FileZilla for Mac users), upload your files, including `regform2.html`, `desktop.css` and `tooltip.css` 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/lab07/regform2.html	http://mercury.swin.edu.au/cos60002/s1234567/lab07/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 10. HTML and CSS Validation

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

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