Facilitator Meeting

Instructions 3 – HTML Forms

Aims

- Use a text editor to create a valid HTML5 form
- Use HTML pattern attributes and other attributes to check the data entered by the user

The tasks are due next week by your workshop class. Tasks are marked in person.

Task: Create a HTML form

Step 1: Create the HTML file structure

Following the procedure set out in previous labs, use a suitable text editor on your local computer (e.g. NotePad++), to create an HTML file called `helpdesk.html` in a `lab03` directory in your working folder. Make sure you include the following elements to define the basic structure and <head> elements:

- At the start of the text file add a document declaration
- Create a `<html>` root element with `<head>` and `<body>` children
- In the `<head>` element create meta-tags for: charset; description; keywords; author
- Give the page a title "CWA Help Desk Appointment" using the appropriate element.

Upload your html to https://validator.w3.org/nu/ to check that is is valid.

Step 2: Add the content to the form

From the file **`lab03.zip`** copy the image file logo.png into your `lab03` folder and view the file **`example_form.html`** in Notepad++. *Using this file as an exemplar*, and referring to other resources (e.g. Lecture notes or W3Schools), create the form shown below (next page). Make sure the content is inside the <body> element you have created in Step 1.

For the `<form> … </form>` element the attributes should be **`method="post"`** and **`action="https://mercury.swin.edu.au/it000000/formtest.php"`**
The `formtest.php` script just echoes back the input data (name = value pairs) sent to the server when the submit button is clicked.

Hint: Implement your submit button `<input type="submit" value="Book"/>` before you create the other elements in your form. Then as soon as you develop other input elements you can test them to see if they are correctly sending their name-value data to the server.

When creating your form keep in mind the following:

- <label>s within the form should link with their <input> controls using a **`for`** attribute that references the **`id`** attribute of the associated input control.
- Use form elements such as <input …/>, <select> … </select> , <text area> … </text area> as appropriate.
- Don't forget to define **`name`** attributes for **all** your inputs otherwise nothing will be sent to the server
- Remember Radio buttons that are in a group all need to share the same **`name`** attribute,
- Define the **`name`** attributes of associated checkboxes as an array.
- Use the **`checked="checked"`** attribute to make the HTML checkbox ticked by default.
- Use <fieldset> … </fieldset> and <legend>…</legend> to appropriate group input controls and labels.
- Use **`placeholder`** attributes on the *Description* `textarea`, and *Date* `input` and *Time* `input` as shown in the illustration on the next page.

`logo.png` (in zip file)
Make this logo hyperlink to
http://www.swin.edu.au
(hint: see Lecture 2 slides)

Heading 1

Paragraphs

Use the **for** attribute of the label to link it to the respective inputs

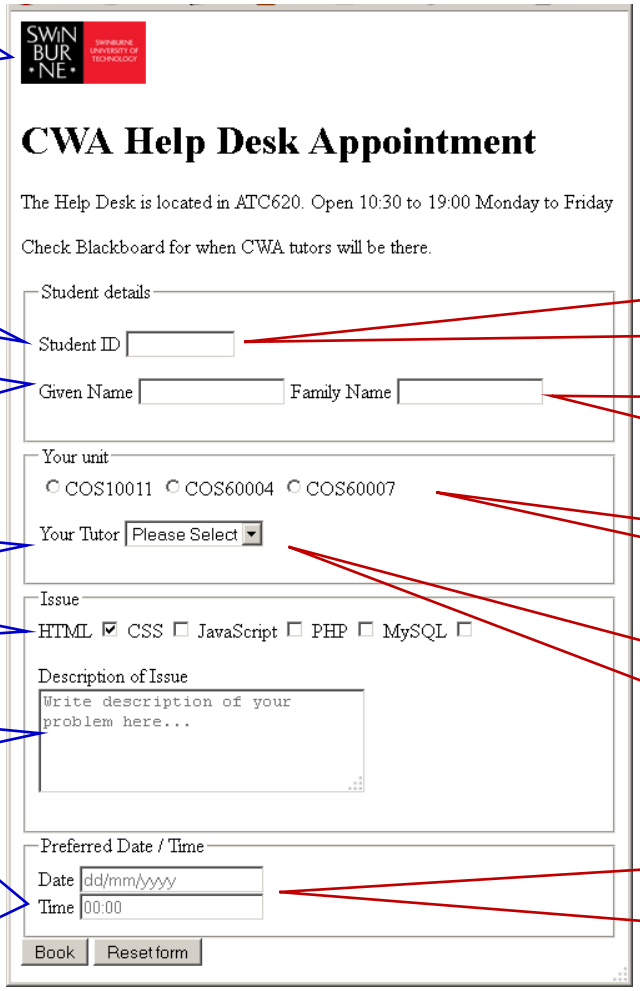
Maximum of 15 characters allowed for given and family names

Make up a list of tutor names as options for your select dropdown list.

Set this checkbox as ticked by default

Use the placeholder attribute to prompt the user

Use <input type="**date**" .../> and <input type="**time**" .../> elements here. *View the Web page in Firefox and in Chrome. What's the difference?*
Change the type to **"text"** and define the placeholder attributes as shown.



Step 3: Data Input Checking

Set the **required** attribute
Use the **pattern** attribute to ensure between 7 and 10 digits are entered.

Given and family names are required. Only alphabetical characters allowed

Set at least one of the radio buttons to required="required"

Make the select element required.
Hint: make the first option value an empty string
<option value="">Please Select</option>

Use the **pattern** attribute to ensure entered data matches the placeholder. Don't worry about using patterns to check the range.

Regularly validate the HTML you are entering by viewing it in Firefox, then right-clicking to view the source to see if any errors are identified.
Upload your html to https://validator.w3.org/nu/ to validate it.

Step 3: Check the input data

Using the Lecture notes as a reference, add **`required`** and **`pattern`** attributes to ensure the data input formats are enforced as defined in the red callout boxes on the right of the figure above.
For example, the time input could be

```
pattern="\d{2}:\d{2}" or better still
pattern="([01]?[0-9]|2[0-3]):[0-5][0-9]"
```

where (a?b|c) is a conditional pattern if a then b else c.

Pattern symbols you might find useful

^	Start of string
$	End of string
.	Match any character
[a-z]	Match the range
\d	Match a digit from 0 – 9
a?	0 or 1 instance of a
a*	0 or more instances of a
a+	1 or more instances of a
a{3}	exactly 3 a's
a{3, }	3 or more a's
a{3,6}	between 3 and 6 a's

Step 4: Upload the html source of the form page to Mercury

Upload the file **helpdesk.html** and **logo.png** to your lab03 folder using WinSCP or similar.
As in the previous tasks view the page in Firefox using an appropriate URL.

Step 5: Revalidate

Upload the html to https://validator.w3.org/nu/ to check it.

[IMPORTANT] Email your facilitator the link to your web page running on the Mercury server by the due date to be marked off. Your work will not be marked without the email.