Lecture 4: Forms

Objectives

- understand how forms work
- use the `<form>` element
- create input objects
- valiate the form on events

element

- container for creating a form
- **usage**: `<form method='post' action='https://a.b.c/register.asp'>`
- **mailto**: `<form action='mailto:joel@abc.com' method='post' enctype='text/plain'>`

Table 1: Form Element Attributes

Attribute	Description
`action`	URL of the application that processes the for data; URL points to a script file or email address.
`enctype`	Content type used to submit the form to the server (when value of method=post); most forms do not need this.
`method`	Specifies HTTP method used to submit the form data; default value is *get*: • get: form data appended to URL specified in `action` attribute • post: form data is sent to server as separate message (more secure)
`accept`	Comma seperated list of content types that a server processing the form can handle correctly; most forms do not need this.
`accept-charset`	List of allowed character sets for input data accepted by the server processing the form; most forms do not need this.

Creating Input Objects

- `<input>` element defines many of the form input object types
- `type` attribute specifies the type of input object
- `required` attribute makes element mandatory to be filled before submitting form

Table 2: Element Types

Type	Description	
`radio`	”Lets user choose one value from a range of values; when radio buttons are grouped together with the same name	only one choice can be selected”
`submit`	Sends form data to server using transmission method specified in the `<form>` element; *every* form needs a submit button	
`reset`	Clears form of any user-entered data and returns form to its original state	
`hidden`	Adds a control that is not displayed in the browser; useful for sending additional information with form data that may be needed for processing	

Type	Description
`image`	Adds a graphic button to the form rather than the default button
`button`	Creates button that has no default behaviour; button function usually defined by a script; when user pushes the button, script function is triggered
`file`	Lets user select a file that is submitted with the form
`range`	Range of values. Shown as a slider control. `<input type='range' min='0' max='10' step='2'>`
`email`	Email address. If email is not syntactically correct, some browsers will display an error message. On mobile, input type can change the virtual keyboard.
`url`	absolute URL value. If URL address is not syntactically correct, some browsers will display an error. On mobile, input type can change the virtual keyboard.
`search`	A search term
`tel`	A telephone number
`color`	A color name; if supported by browser, input type will display color picker that lets user choose a color value from a color wheel or chart.

Labeling Form Elements

- `<label>` element lets you create a caption for an input element
- lets you extend clickable area of a form element
- to make text clickable, associate `<label>` with the `<input>` element by using the `for` and `id` attributes

```
<p>
<label for='First Name'>First Name:</label>
<input type='text' required name='firstname' id='First Name' size='35' maxlength='35' />
</p>
```

Creating Text Boxes

```
<input type='text' name='firstname' size='20' maxlength='35' value='First Name' />
```

Creating Check Boxes

- check boxes are an on/off toggle that the user can select

```
<input type='checkbox' name='species' value='smbass'> Smallmouth Bass
```

Creating Radio Buttons

- radio buttons are like checkboxes, but only one selection is allowed

```
<p>Would you like to be on our mailing list?</p>
<p>
    <input type='radio' name='list' value='yes' id='Yes'>
    <label for='Yes'>Yes</label>
    <input type='radio' name='list' value='no' id='No'>
    <label for='No'>No</label>
</p>
```

Creating an Image for the Submit Button

- choose an image file and use it instead of the default submit button

```
<input type='image' src='submit.gif' alt='submit button'>
```

Letting User Submit a File

```
<p>Use the browse button to select your file:</p>
<p><input type='file' size='30'></p>
```

Password Entry Field

- like text input, but entered text is hidden by asterisks

```
<p>Ennter username and password</p>
<p>
    User Name: <input type='text' size='30' />
    Password: <input type='password' size='30' />
</p>
```

Element

- choose to let user pick multiple values from the list by adding the multiple attribute

```
<select name='snacks' multiple size='6'>
    <option>chips</option>
    <option>nachos</option>
    ...
</select>
```

- group and label sets of list options with the element and label attribute

```
<optgroup label='Salty Snacks'>
    <option>Potato Chips</option>
</optgroup>
```

```
<textarea> Element
------------------
* create a larger text area for user input
```

Briefly tell us your favourite fish story:

Creating Input Groupings

- use `<fieldset>` and `<legend>` elements to create groupings of different types of input elements

Forms Events

Table 3: Form Events

Attribute	**Value**	**Description**
`onblur`	*script*	Fires the moment the element loses focus
`onchange`	*script*	Fires the moment the value of the element is changed
`oncontextmenu`	*script*	Script to be run when a context menu is triggered
`onfocus`	*script*	Fires the moment the element gets focus
`oninput`	*script*	Script to be run when an element gets user input
`oninvalid`	*script*	Script to be run when an element is invalid
`onreset`	*script*	Fires when the Reset button in a form is clicked
`onsearch`	*script*	Fires when user writes something in a search field (for <input=‘search’>)
`onselect`	*script*	Fires after some text has been selected in an element
`onsubmit`	*script*	Fires when a form is submitted

Keyboard Events

Table 4: Keyboard Events

Attribute	**Value**	**Description**
`onkeydown`	*script*	Fires when a user is pressing a key
`onkeypress`	*script*	Fires when a user presses a key
`onkeyup`	*script*	Fires when a user releases a key

Mouse Events

Table 5: Mouse Events

Attribute	**Value**	**Description**
`onclick`	*script*	Fires on a mouse click on the element
`ondblclick`	*script*	Fires on a mouse double-click on the element
`onmousedown`	*script*	Fires when a mouse button is pressed down on an element
`onmousemove`	*script*	Fires when the mouse pointer is moving while it is over an element
`onmouseout`	*script*	Fires when the mouse pointer moves out of an element
`onmouseover`	*script*	Fires when the mouse pointer moves over an element
`onmouseup`	*script*	Fires when a mouse button is released over an element
`onwheel`	*script*	Fires when the mouse wheel rolls up or down over an element

Summary

- choose the right form elements based on the data you want to collect
- form element has attributes that describe how the form data is processed
- need a server application to process form data
- `<fieldset>` and `<legend>` elements let you create more visually appealing forms