Tutorial 7: Designing a Web Form

Hia Al Saleh

November 5th, 2024

Contents

1 Objectives

- Explore web forms
- Work with form servers
- Create forms and field sets
- Create labels and input boxes
- Explore form layout
- Work with date and time fields
- Create a selection list
- Create option buttons
- Create check boxes and text area boxes
- Create spinners and range sliders
- Use data lists
- Create form buttons
- Validate a form
- Apply validation styles

2 Structure of a Web Form

2.1 Introducing Web Forms

A web form allows users to enter data that can be saved and processed. It is a common way to accept user input and allows the creation of interactive websites for user feedback.

2.2 Parts of a Web Form

Controls, also known as widgets, are the objects that allow a user to interact with a form. Each data entry control is associated with a data field, which stores the data values supplied by a user.

2.3 Types of Controls

- Input boxes to insert text and numeric values
- Option/radio buttons to select data values from a predefined set of options
- Selection lists to select data values from an extensive list of options
- Check boxes to select data values limited to two possibilities, such as "yes" or "no"
- Text area boxes to enter text strings that may include several lines of content

2.4 Types of Widgets

- Spin boxes to enter integer values confined to a specified range
- Slider controls to enter numeric values confined to a specified range
- Calendar controls to select date and time values
- Color pickers to choose color values

3 Forms and Server-Based Programs

Field values entered by a user are processed by a program running on the user's computer or on a web server in a secure location. For example, a web form is used to collect data from a customer for an order, and the server program processes the data and handles the billing and delivery.

4 Starting a Web Form

Web forms are marked using the form element:

```
<form id="text" attributes>
    content
</form>
```

where:

- id identifies the form
- attributes specify how the form should be processed by the browser
- content is the form's content

A form element can be placed anywhere within the body of a page and can also contain page elements such as tables, paragraphs, inline images, and headings.

5 Interacting with the Web Server

The action, method, and enctype attributes have to be included in a form to specify where and how to send the form data:

```
<form action="url" method="type" enctype="type">
    content
</form>
```

where:

- action attribute provides the location of the web server program that processes the form
- method attribute specifies how the browser should send form data to the server
- enctype attribute specifies how the form data should be encoded as it is sent to the server

5.1 Method Attribute Values

There are two possible values for the method attribute:

- Get method: Tells the browser to append the form data to the end of the URL specified in the action attribute. The get method is the default method.
- Post method: Sends the form data in its own separate data stream. The post method is considered to be a more secure form of data transfer.

5.2 Enctype Attribute Values

The enctype attribute has three possible values that determine how the form data is encoded when sent to the server.

6 Creating a Field Set

A field set groups fields that share a common purpose. Field sets are created using the fieldset element:

```
<fieldset id="id">
    content
</fieldset>
```

where id identifies the field set and content is the form content within the field set.

6.1 Adding a Field Set Legend

The legend describes the content of a field set using the legend element:

```
<legend>text</legend>
```

where text is the text of the legend. The legend element contains only text and no nested elements. By default, legends are placed in the top-left corner of the field set box and can be moved to a different location using CSS positioning styles.

7 Creating Input Boxes

The syntax for the input element is:

```
<input name="name" id="id" type="type" />
```

where:

- The name attribute provides the name of the data field associated with the control.
- The id attribute identifies the control in which the user enters the field value.
- The type attribute indicates the data type of the field.

7.1 Input Types and Virtual Keyboards

Virtual keyboards are software representations of a physical device. Web forms can be made responsive by displaying different virtual keyboards for each input type. For example, an input box for a telephone number is more convenient to read with digits displayed prominently on the keyboard.

8 Adding Field Labels

To associate a text string with a control, the text string has to be enclosed within the label element:

```
<label for="id">label text</label>
```

where id is the id of the control that is associated with the label and label text is the text of the label.

9 Designing a Form Layout

There are two general layouts for forms:

- Labels are placed directly above the input controls.
- Labels and controls are placed side-by-side.

9.1 Defining Default Values and Placeholders

The value attribute is used to specify a default field value. Placeholders are text strings that appear within a form control, providing a hint about the kind of data that should be entered into a field. They are defined using the placeholder attribute:

```
placeholder="text"
```

where text is the text of the placeholder.

10 Entering Date and Time Values

Date and time fields ensure that users enter data in the correct format, indicated using type attributes: date, time, datetime-local, month, and week.

11 Creating a Selection List

A selection list is a list box that presents users with a group of possible values for the data field. The list is created using the select and option elements:

```
<select name="name">
    <option value="value1">text1</option>
    <option value="value2">text2</option>
    ...
</select>
```

where name is the name of the data field, value1, value2,... are the possible field values, and text1, text2,... are the text of the entries in the selection list that users see on the web form.

11.1 Working with Select Attributes

By default, a selection list appears as a drop-down list box. To display a selection list as a scroll box, use the size attribute in the select element:

```
<select size="value"> ... </select>
```

where value is the number of options that the selection list displays at one time.

11.2 Allowing Multiple Selections

By default, selection lists allow only one selection from the list of options. To allow more than one item to be selected, add the multiple attribute:

```
<select multiple> ... </select>
```

11.3 Grouping Selection Options

Organize selection list options by placing them in option groups using the optgroup element:

```
<select>
    <optgroup label="label1">
        <option>text1</option>
        <option>text2</option>
    </optgroup>
</select>
```

where label1 is the label for the different groups of options.

12 Creating Option Buttons

Option buttons, also called radio buttons, allow users to select one option from a set. They are created with a group of input elements with a type attribute value of ”radio”:

```
<input name="name" value="value1" type="radio" />
<input name="name" value="value2" type="radio" />
```

where name is the name of the data field and value1, value2, etc., are the field values associated with each option. Set an option button to be selected as the default by adding the checked attribute:

```
<input name="name" type="radio" checked />
```

13 Creating Check Boxes

Check boxes are designed for fields that record the presence or absence of an object or event. They are created using the input element with the type attribute set to ”checkbox”:

```
<input name="name" value="value" type="checkbox" />
```

where the value attribute contains the value of the field when the check box is checked.

14 Creating a Text Area Box

A text area is created using the textarea element:

```
<textarea name="name">text</textarea>
```

where text is the default value of the data field. HTML supports the rows and cols attributes to set the text area size:

```
<textarea rows="value" cols="value">...</textarea>
```

where rows attribute specifies the number of lines in the text area box and cols attribute specifies the number of characters per line.

15 Entering Numeric Data

15.1 Creating a Spinner Control

A spinner control displays an up or down arrow to increase or decrease the field value by a set amount. To create a spinner control, apply the input element using the number data type:

```
<input type="number" />
```

15.2 Creating a Range Slider

A slider control limits a numeric field to a range of possible values. To create a slider control, apply the range data type in the input element:

```
<input type="range" />
```

16 Suggesting Options with Data Lists

A data list is a list of possible data values that a form field can have. Data lists are defined using the datalist element:

```
<datalist id="id">
    <option value="value" />
    <option value="value" />
</datalist>
```

17 Working with Form Buttons

Form buttons are a type of form control that performs an action, such as submitting the form to a program running on the web server or resetting the form fields to their default values.

17.1 Creating a Command Button

A command button runs a program that affects the content of a page or the actions of a browser. It is created using the input element with the type attribute set to button:

```
<input value="text" onclick="script" type="button" />
```

17.2 Creating Submit and Reset Buttons

A submit button submits a form to the server for processing when clicked:

```
<input value="text" type="submit" />
```

A reset button resets a form, changing all fields to their default values:

```
<input value="text" type="reset" />
```

17.3 Designing a Custom Button

For more control over a button's appearance, use the button element:

```
<button type="text">content</button>
```

18 Validating a Web Form

Validation is the process of ensuring that a user has supplied valid data. There are two types of validation:

- Server-side validation – validation occurs on the web server.
- Client-side validation – validation occurs in the user's browser.

18.1 Identifying Required Values

To verify if data is supplied for all required data fields, add the required attribute to the control. If a required field is left blank, the browser will not submit the form, returning an error message.

18.2 Validating Based on Data Type

A form fails the validation test if the data values entered into a field do not match the field type. For example, a data field with the number type will be rejected if non-numeric data is entered. Fields with email and URL types will be rejected if a user provides an invalid email address or text that does not match the format of a URL.

18.3 Testing for a Valid Pattern

To test whether a field value follows a valid pattern of characters, test the character string against a regular expression. A regular expression or `regex` is a concise description of a character pattern. To validate a text value against a regular expression, add the `pattern` attribute to the input element:

```
<input name="custZip" pattern="^\d{5}$" />
```

where the regular expression `^\d{5}$` represents any string of five numeric characters.

18.4 Defining the Length of the Field Value

The syntax to define the maxlength attribute is:

```
<input maxlength="value" />
```

where value is the maximum number of characters in the field value. For example:

```
<input name="custZip" maxlength="5" />
```

The maxlength attribute does not distinguish between characters and digits.

18.5 Applying Inline Validation

Inline validation is the technique of immediate data validation and reporting of errors. The focus pseudo-class is used to change the display style of fields that currently contain invalid data. Focus is the state in which an element has been clicked by the user, making it the active control on the form.

18.6 Pseudo-Classes for Valid and Invalid Data

The valid and invalid pseudo-classes are used to format controls based on whether their field values pass or fail a validation test. For example, to display input elements containing valid data with a light green background, use the following style rule:

```
input:valid {
    background-color: rgb(220, 255, 220);
}
```

To display input elements containing invalid data with a light red background when focused, use:

```
input:focus:invalid {
    background-color: rgb(255, 232, 233);
}
```