

SYST10199

Web Programming Foundations

HTML Forms. Client-Side Validation. Styling HTML Forms

Agenda

- Understanding HTML Forms
- Basic Form Input Elements
- Other Important Form Elements
- Client Side Validation
- Styling HTML Forms
- Exercises

Understanding HTML Forms

- HTML forms are used to collect user input, such as names, email addresses, passwords, and other data.
 - This information is then sent to a server for processing.
- Forms are essential for many web functions:
 - User registration and login
 - Collecting user credentials to create and manage accounts.
 - Contact forms
 - Allowing visitors to send messages or inquiries.
 - E-commerce
 - Gathering shipping information and payment details for online purchases.
 - Search bars
 - Enabling users to search for content on a website.

Structure of an HTML Form

- An HTML form is created using the `<form>` element.
 - It is the container for all the other form input elements.
- The term **form input elements** refers to all the various HTML elements that are used to collect different types of inputs from the user.
 - Some of these are referred to as `<input>` elements
 - Others have different names (`<select>`, `<textarea>` etc...)
- The basic structure of a form is:

```
<form name="quiz" action="submit_form.js" method="POST">  
    <!-- Form elements go here -->  
</form>
```

The name Attribute

- The name attribute of an HTML form is used to identify the form when it is submitted.
- It serves several important purposes:
 - 1. Identification:**
 - The name attribute allows the form to be uniquely identified in scripts or when processing the form data on the server side.
 - Helpful when multiple forms are present on a single page.
 - 2. Data Submission:**
 - Data is sent to the server with the form's name as part of the request.
 - Helps the server-side script to recognize which form's data is being processed.
 - 3. JavaScript Access:**
 - The name attribute can be used in JavaScript to easily access and manipulate the form or its elements.

The `action` Attribute

- The `action` attribute specifies the URL to which the form data will be sent when the form is submitted.
- Plays a crucial role in determining how the data is processed
 - It indicates the name of the script or program that will process your form's input data.
- If your form is processed on the server, then the `action` attribute will contain the name of a file on the server.
- When the user submits the completed form, your browser will make a new HTTP request to the file named in the `action` attribute.
 - `action="submit_form.js"` indicates that when the form is submitted, the browser will make a new Http request to the file `"submit_form.js"`.
 - If you leave out the `action` attribute, then the form will be processed by the "current page"

The method Attribute

- Specifies the HTTP method that should be used when submitting the form data to the server.
- The most used values for the method attribute are GET and POST.
 - GET Method:
 - Appends the form data to the URL as query parameters. This means that the data is visible in the URL.
 - Used for retrieving data or when the form submission does not change the server state.
 - The amount of data that can be sent is limited by the maximum URL length
 - Sensitive information should not be sent using this method since it is visible in the URL.
 - POST Method:
 - Sends the form data in the body of the HTTP request, keeping it hidden from the URL.
 - It is used for submitting data that changes the server state
 - There is no limit on the amount of data that can be sent
 - It is also more secure for transmitting sensitive information.

Web Programming

HTML Form Elements

The `<input>` Element

- The `<input>` element is the most versatile form control.
- The type attribute determines the type of input field to be displayed.
- Some of the most common type values are:
 - `text` : A single-line text input field.
 - `password`: A text input field where the characters are masked
 - `email`: An input field for an email address.
 - The browser provides basic validation.
 - `number`: An input field for a number.
 - `radio`: A radio button, used for selecting one option from a set of choices.
 - `checkbox`: A checkbox, used for selecting zero or more options.
 - `submit`: A button that submits the form data to the server.
 - `reset`: A button that resets the form.
 - `date`: A control for entering a date.

<input> Element Attributes

- The <input> tag has a set of attributes that define what the input control should look like and how it should work.
- Some of these attributes are:
 - type="":
 - Defines what type of input field this is (e.g., text, password, radio)
 - name="":
 - The name of the input field; it is used when form data is submitted to the server.
 - value="":
 - The value of the control, as entered by the user.
 - You can assign a default value to the value attribute in your HTML code.
 - What the value attribute will be, depends on the type attribute.
 - size="n":
 - The visible size of the field, measured by number of characters (based on font size)
 - Note that elements can also be sized using CSS.

Other Important Form Elements

- Besides `<input>` element, forms can have other elements such as:
 - `<label>`:
 - Provides a descriptive label for an input field. It's good practice to wrap the input inside the label or use the `for` attribute to link the label to the input's id. This improves accessibility.
 - `<textarea>`:
 - Used for multi-line text input (e.g., a message or comment).
 - `<select>` and `<option>`:
 - Used to create a dropdown list. The `<select>` element is the container, and each `<option>` element represents a choice in the list.
 - `<button>`:
 - Used for generic buttons

The <label> Element

- It is usually used to indicate to the user input is required in each field.
- For example:

```
<label for="userName">Username:  
  <input type="text" name="userName">  
</label>  
<label for="password">Password:  
  <input type="password" name="password">  
</label>
```

- Will display:

User Name: Password:

for Attribute of <label> Element

- The for attribute in the <label> element is used to associate the label with a specific form control, such as an input field.
- It enhances accessibility and improves user experience in several ways:
 - **Accessibility:**
 - Screen readers can provide better context to users with visual impairments.
 - When a user focuses on the label, the corresponding input field is also highlighted, making navigation easier.
 - **Clicking the Label:**
 - Clicking on the label will focus or activate the associated input field.
 - Particularly useful for checkboxes and radio buttons.
 - **Improved Usability:**
 - Helps users understand what information is required in each field, leading to a more intuitive form-filling experience.

The <textarea> Element

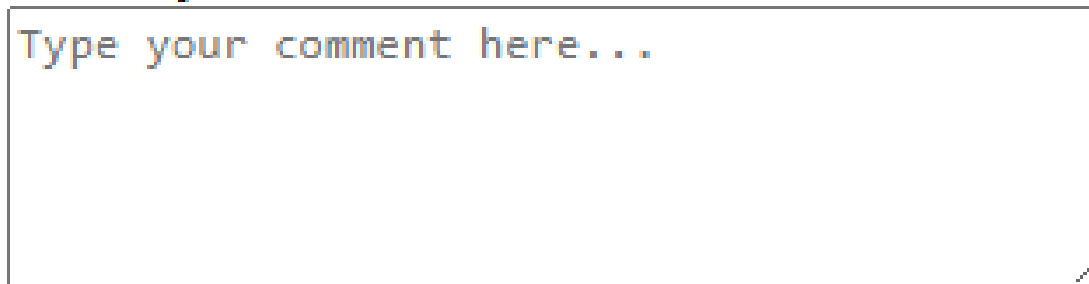
- It is used to create a multi-line plain-text editing control.
- Ideal for collecting longer user inputs
- It can have the following attributes:
 - **name:**
 - Required to identify the data when it's submitted.
 - **id:**
 - Required to link the <textarea> to a <label>.
 - **rows:**
 - Specifies the visible height of the text area in lines.
 - **cols:**
 - Specifies the visible width of the text area in average character widths.
 - **placeholder:**
 - Provides a hint to the user about what to enter in the field.
 - The text disappears when the user starts typing.

<textarea> Example

```
<label for="comment_box">Leave your comment here:</label><br>
<textarea id="comment_box" name="user_comment"
  rows="5" cols="40"
  placeholder="Type your comment here...">
</textarea>
```

- It will display:

Leave your comment here:



Type your comment here...

The `<select>` and `<option>` Elements

- The `<select>` and `<option>` elements create a dropdown list.
- Nice way to present users with a predefined set of choices without taking up a lot of space.
- `<select>`:
 - The container for the dropdown menu.
 - It must have a name and an id to be properly linked with a label.
 - `size` attribute can be used in `<select>` to define how many options will be visible.
- `<option>`:
 - Represents a single choice within the dropdown list.
 - The text inside the tag is what the user sees, and the `value` attribute is the data that will be sent to the server if that option is selected.
 - `selected` attribute can be added to one of the option elements to make it the default choice when the page loads.

<select> and <option> Example – No size

```
<label for="pet-select">Choose a pet:</label>
<select name="pets" id="pet-select">
  <option value="">--Please choose an option--</option>
  <option value="dog">Dog</option>
  <option value="cat">Cat</option>
  <option value="hamster">Hamster</option>
  <option value="parrot">Parrot</option>
  <option value="spider">Spider</option>
  <option value="goldfish">Goldfish</option>
</select>
```

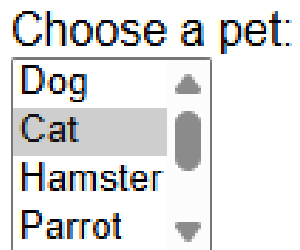
- Will display:

Choose a pet:

Example 2 – Using size and selected

```
<label for="pet-select">Choose a pet:</label><br>
<select name="pets" id="pet-select" size="4">
  <option value="dog">Dog</option>
  <option value="cat" selected>Cat</option>
  <option value="hamster">Hamster</option>
  <option value="parrot">Parrot</option>
  <option value="spider">Spider</option>
  <option value="goldfish">Goldfish</option>
</select>
```

- Will display:



Fieldset and Legend Elements

- Sometimes it's helpful to visually group your controls on the page.
- It can be done using `<fieldset>` and `<legend>` elements
- Fieldsets are containers that can contain a group of related controls
- They aid users with cognitive disabilities by helping them understand the relationship the inputs have with each other and the rest of the inputs on the screen.
- They also helps users with screen readers when they encounter groups of controls like radio buttons and check boxes, by associating the controls together with the text inside the `<legend>` element.

Fieldset and Legend Example

```
<fieldset style="width: 20%; padding: 0px .5em">
  <legend style="font-weight: bold; padding: .4em 0px"> Favourite Colour
</legend>
  <div><label for="blue" style="color: blue;">
    <input type="radio" name="faveColour" value="blue" id="blue">
    Blue</label>
  </div>
  <div><label for="green" style="color: green;">
    <input type="radio" name="faveColour" value="green" id="green">
    Green</label>
  </div>
  <div><label for="red" style="color: red;">
    <input type="radio" name="faveColour" value="red" id="red">
    Red</label>
  </div>
</fieldset>
```

- It will display:

Favourite Colour

☐ Blue

☐ Green

☐ Red

Exercise 2.1 – Part A and B

- Follow instructions to create a registration Form

Web Programming

Client-Side Validation

Data Validations

- Forms are used to collect user inputs, which will be processed.
- In most applications, the processing happens on the server, using server-side technologies such as PHP, Java Enterprise (servlets), Node.js etc..
- A crucial aspect of handling form input data is validating it to ensure accuracy and that it falls within specified ranges or values.
- While this validation can be performed on the server side, it is advisable to validate input data on the client side before it reaches the server.
- Client-side validation can be achieved using JavaScript, but thanks to recent enhancements in HTML, we can perform a significant amount of validation directly through HTML attributes on our input fields.

What To Validate?

- Input data validation generally involves checking for the following:
 - Ensuring an input value is of the required type.
 - A numeric field can only contain digits, or making sure a decimal field only contains digits and a, optional, single decimal point.
 - A text field cannot contain any characters that aren't letters and spaces.
 - Ensuring that mandatory input fields aren't left empty.
 - Ensuring input data is in the right format.
 - A Canadian postal Code must be in the form A1B 2C3.
 - A student ID must be exactly 9 digits.
 - A person's username or login name must be between 8 and 16 characters and must contain only letters, digits, and the underscore character.
 - Ensuring data is within valid ranges.
 - A price must be a positive number.
 - A province name must be one of the existing provinces.
 - A date must be valid date (e.g., June 31, August 42, and Feb 29, 2001, would be invalid).

Why Client-Side Data Validations

- Data validation is essential for several reasons:
 1. Ensures Data Accuracy
 - Validating data helps ensure that the information entered by users is accurate
 2. Prevents Security Vulnerabilities
 - By validating inputs, you can prevent malicious data from being processed by the server.
 3. Enhances User Experience
 - When users receive immediate feedback on their input errors, it improves their experience.
 4. Maintains Data Integrity
 - Validating data helps maintain the integrity of the database by ensuring that only valid and relevant data is stored.
 5. Reduces Processing Load
 - By validating data on the client side, you can reduce the load on the server
 6. Compliance with Business Rules
 - Data validation ensures that the input adheres to specific business rules and requirements.

Validating the Data Type

- HTML5 introduced several input types that facilitate data type validation:
 1. Text: General text input.
 2. Email: Valid email addresses.
 3. Number: Numeric values.
 4. URL: Valid web addresses.
 5. Date: Dates in a specific format.
- Using the appropriate input types helps browsers enforce validation rules automatically.

Mandatory (Required) Fields

- HTML5 introduced the [required attribute](#), which can be added to input fields to indicate that they must be filled out before the form can be submitted.
- Example:

```
<form action="submit_form.js" method="post">  
  <label for="field">Required Field:</label>  
  <input type="text" id="username" name="field" required><br>  
  <input type="submit" value="Submit">  
</form>
```
- The form will display the following if user clicks the submit button, without entering anything:

Required Field:

 Please fill out this field.

Formats and Patterns

- Often you will need to check if the input follows a certain pattern:
 - Verifying that a text field contains only letters.
 - Ensuring a numeric field consists solely of digits.
 - Confirming that a field contains a Canadian Postal Code formatted as A1B 2C3, with or without a space.
 - Validating that a course code consists of 4 letters followed by 5 digits.
- This can be accomplished using [pattern attribute](#)
 - It accepts a [regular expression](#), which is a sequence of special characters used for "pattern matching."
 - For example, the regular expression `\d{9}` matches exactly 9 digits.

Pattern Example

```
<form action="submit_form.js" method="post">
  <label for="student">Student Name:</label>
  <input type="text" id="stName" name="student" required><br>

  <label for="studentNo">Student No:</label>
  <input type="text" id="stNo" name="studentNo" pattern="\d{9}" required><br>

  <label for="postal">Postal Code:</label>
  <input type="text" id="postCode" name="postal" pattern="[A-Z][0-9][A-Z] ?[0-9][A-Z][0-9]" required><br>

  <label for="course">Course Code:</label>
  <input type="text" id="courseCode" name="course" pattern="[A-Z]{4}\d{5}"
required><br>

  <input type="submit" value="Submit">
</form>
```

The Range of Data

- There are two sets of attributes that lets you specify the range or the minimums and maximums for a field:
 - The minimum or maximum of characters allowed in the field:
 - `minlength=""` accepts an integer number that defines the minimum number of characters permitted in a field.
 - If the number of characters is less than `minlength`, an error message appears on form submission.
 - `maxlength=""` accepts an integer number that defines the maximum number of characters permitted in a field.
 - If the number of characters is greater than `maxlength`, an error message appears on form submission.
 - The minimum or maximum value allowed in a numeric field:
 - `min=""` accepts an integer number and defines the minimum value permitted in a numeric field.
 - If the input value is less than `min`, an error message appears on form submission.
 - `max=""` accepts an integer number and defines the maximum value permitted in a numeric field.
 - If the input value is greater than `max`, an error message appears on form submission.
 - The `min` and `max` attributes can be used on `<input>` elements that have a `type=""` of `number`, `range`, and several of the date/time input types.

The Range of Characters in a Field

- The following is an example of a field that accepts only 4 to 10 characters:

```
<form action="submit_form.js" method="post">  
  <label for="field">Must be between 4 to 10 characters:</label>  
  <input type="text" id="fieldwithLength" name="field" required minlength="4"  
    maxlength="10"><br>  
  
  <input type="submit" value="Submit">  
</form>
```

- The above code will display:

Must be between 4 to 10 characters:

- If the user enters less than four characters and clicks "submit", an error message will be shown requiring four characters or more.
- The user will also not be able to enter more than 10 character as the field after the 10th character will not be accepted.

The Range of Numeric Values

- The following numeric field accepts a positive number that is less than 100:
 - `<form action="submit_form.js" method="post">`
 - `<label for="numRange">Must be between 4 to 10 characters:</label>`
 - `<input type="number" id="nummbers" name="numRange" required min="0" max="100">
`
 - `<input type="submit" value="Submit">`
 - `</form>`
- The code above will be displayed as:

Must be between 4 to 10 characters:
- If the user attempts to enter a number that is less than 0 or greater than 100, an error message will be displayed.

Exercise 2.1 – Part C

- Follow instructions to validate the data the user can enter in the form

Web Programming

Styling HTML Forms

Styling HTML Forms

- You may have encountered forms on websites that offer visual indicators for form validation.
- For instance, an icon might appear, or the background or border of an input field may change color when the input is deemed invalid.
- This effect is achieved using specific CSS pseudo-classes, which you have learned in Web Development course.
- The pseudo classes that will be helpful in providing visual feedback for form validation are:
 - [:valid](#) – is used to style an element that contains a valid input
 - [:invalid](#) – is used to style an element that contains an invalid input;
 - [:required](#) – is used to style an element with the required attribute

Other Form Pseudo-Selectors

- You already know how to style an element using the `:hover` pseudo class.
 - It allows you to create visually pleasing affects on buttons and other form controls.
- To make the form more user-friendly, you can use focusable elements
 - Pseudo classes `:focus` and `:focus-visible` are used for this purpose
- The element that has focus will respond to any key presses.
 - If a text field is focused, any keys you press will result in the corresponding characters appearing in that field.
 - If a checkbox or radio button is focused, pressing the SPACE bar will toggle its selection state.
 - If a button is focused, pressing ENTER will act as if the button has been clicked with the mouse.

:focus and :focus-visible

- The [:focus pseudo class](#) is used to style the appearance of an element when it has the focus.
 - An item receives the focus when the user tabs into it or on it, or when the user clicks in or on the item.
- The [:focus-visible](#) pseudo class is also used to style the appearance of an element when it has the focus.
- The difference between :focus and :focus-visible is that :focus-visible styles apply only when the user needs to be informed that the element has the focus.
 - When the user clicks on a button, there is no need to indicate that the button has the focus.
 - If the user tabs to the button, they need to know that they tabbed to the right control
 - :focus-visible will only style the button when the user tabs to the button.

Styling Active Elements

- You can also style controls that are active:
 - An active control is in the process of being clicked, so this generally only applies to links and buttons.
 - can be done by using the [:active pseudo class](#).
- Example:

HTML	CSS
<pre><form> <p>* indicates a required field</p> <p> <label for="user">User Name: * <input type="text" id="user" required> </label> </p> <p><input type="submit" value="Submit"></p> </form></pre>	<pre>input[type='submit']:active { box-shadow: 0px 0px .2em .2em pink; }</pre>

Styling Checked/Selected Items

- You can apply styles to items that are in the checked or selected state by utilizing the [:checked](#) pseudo-class.
- It allows customization the appearance of checkboxes and radio buttons once they have been checked or selected.
- While browsers provide default styling for these states, you have the flexibility to modify their appearance using the [:checked](#) pseudo-class.
- CSS Example:

```
input[type="radio"]:checked,  
input[type="checkbox"]:checked {  
    box-shadow: 0px 0px .2em .2em #C8E6C9;  
}
```

```
option:checked {  
    background-color: teal;  
    color: white;  
}
```

Placeholders

- Placeholders are text contents that are often added to text input fields to add additional information about the required or expected inputs
- They can also be used as field label when there isn't enough screen real estate for an actual `<label>` element.
- For example:

```
<label for="fname">First Name:  
  <input type="text" id="fname" name="firstname" placeholder="first or given name">  
</label>
```
- Will display:

First Name:

Styling Placeholders

- The [:placeholder-shown](#) pseudo class is used to style the appearance of an element when it contains placeholder text.
- The [::placeholder](#) pseudo element is used to style the actual placeholder text.
- Example:

```
/* placeholder styling: */  
input:placeholder-shown {  
    outline: 0.15em solid #c2c2d6;  
}
```

```
input::placeholder {  
    color: #8585ad;  
}
```

Exercise 2.1 – Part D

- Follow instructions to style the form and make it look like the image

Next Class

- Read the material posted on slate as well as the links and resources
- Get ready for Quiz 1
 - Review the lessons 1 and 2
- Assignment 1 is posted
 - Due next week

Reference

- **Code Guide**

- Link: [Code Guide](#)
- Description: A comprehensive guide to best practices in coding, focusing on HTML, CSS, and JavaScript.

- **Fundamentals of Web Programming**

- Link: [Fundamentals of Web Programming](#)
- Description: An interactive textbook covering the essential concepts of web programming, including HTML, CSS, and JavaScript.