Microsoft® Official Course

Module 4

Creating Forms to Collect and Validate User Input

Module Overview

- Creating HTML5 Forms
- Validating User Input by Using HTML5 Attributes
- Validating User Input by Using JavaScript

Lesson 1: Creating HTML5 Forms

- Declaring a Form in HTML5
- HTML5 Input Types and Elements
- HTML5 Input Attributes

Declaring a Form in HTML5

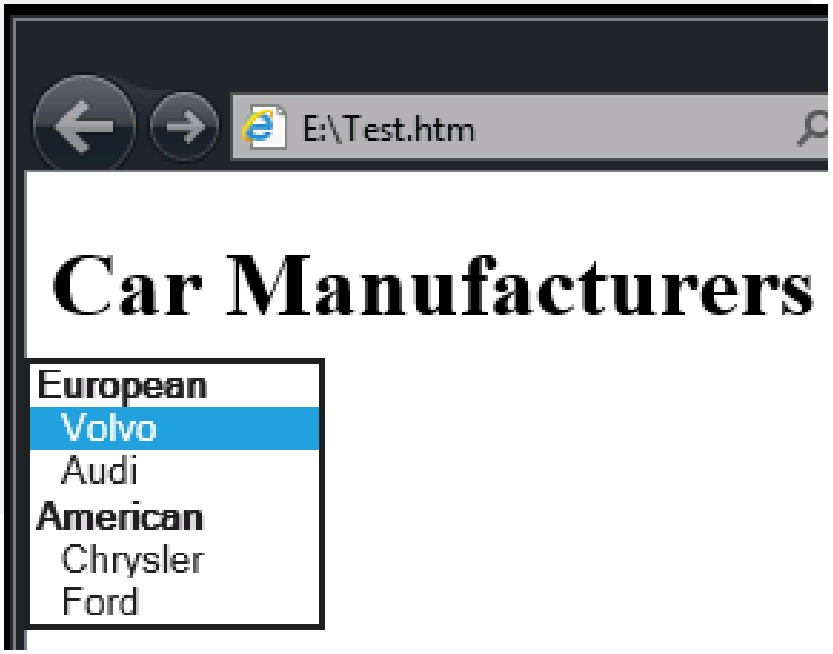
- Use an HTML5 form to gather user input:

```
<form name="userLogin" method="post" action="login.aspx">
  <fieldset>
    <legend>Enter your log in details:</legend>
    <div id=”usernameField” class="field">
      <input id="uname" name="username" type="text"
         placeholder="First and Last Name" />
      <label for="uname">User's Name:</label>
    </div>
    <div id="passwordField" class="field">
      <input id="pwd" name="password" type="password"
         placeholder="Password" />
      <label for="pwd">User's Password:</label>
    </div>
  </fieldset>
  <input type="submit" value="Send" />
</form>
```

HTML5 Input Types and Elements

- HTML5 defines a wide range of new input types and elements, but not all are widely implemented

```
<select id="carManufacturer" name="carManufacturer">
  <optgroup label="European">
    <option value="volvo">Volvo</option>
    <option value="audi">Audi</option>
  </optgroup>
  <optgroup label="American">
    <option value="chrysler">
        Chrysler</option>
    <option value="ford">
        Ford</option>
  </optgroup>
</select>
```



Car Manufacturers

```
European
  Volvo
  Audi
American
  Chrysler
  Ford
```

HTML5 Input Attributes

- Input attributes modify the behavior of input types and forms to provide better feedback and usability:
  - autofocus
  - autocomplete
  - required
  - pattern
  - placeholder
  - many other input type-specific attributes

Lesson 2: Validating User Input by Using HTML5 Attributes

- Principles of Validation
- Ensuring that Fields are Not Empty
- Validating Numeric Input
- Validating Text Input
- Styling Fields to Provide Feedback

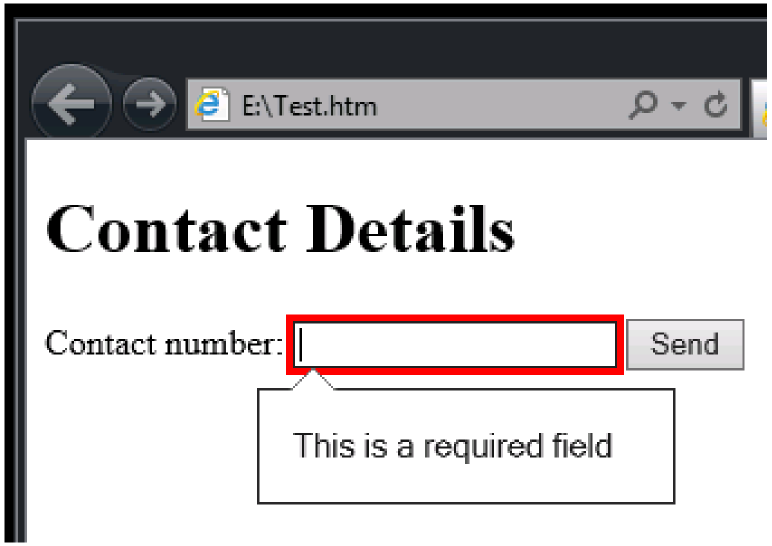
Principles of Validation

- User input can vary in accuracy, quality, and intent
- Client-side validation improves the user experience
- Server-side validation is still necessary

Ensuring that Fields are Not Empty

- Use the **required** attribute to indicate mandatory fields
  - The browser checks that they are filled in before submitting the form

```
<input id="contactNo" name="contactNo" type="tel"
placeholder="Enter your mobile number" required="required" />
```



Contact Details

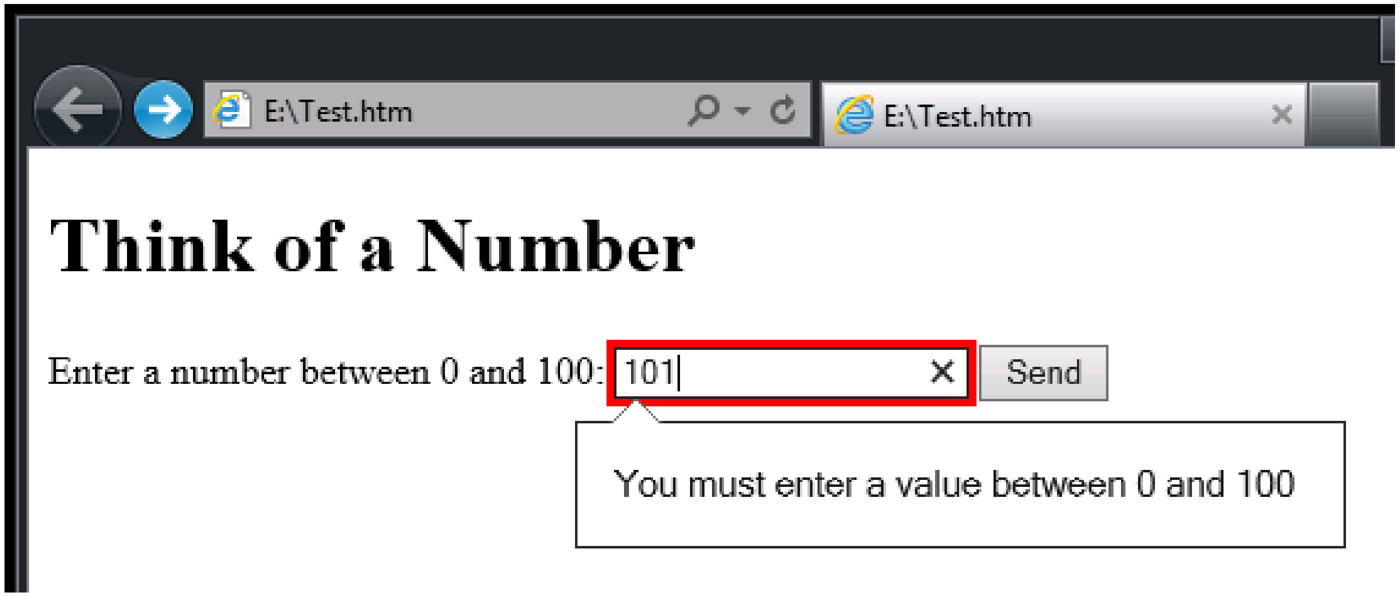
Contact number:

This is a required field

Validating Numeric Input

- Use the **min** and **max**attributes to specify the upper and lower limit for numeric data

```
<input id="percentage" type="number" min="0" max="100" />
```



Think of a Number

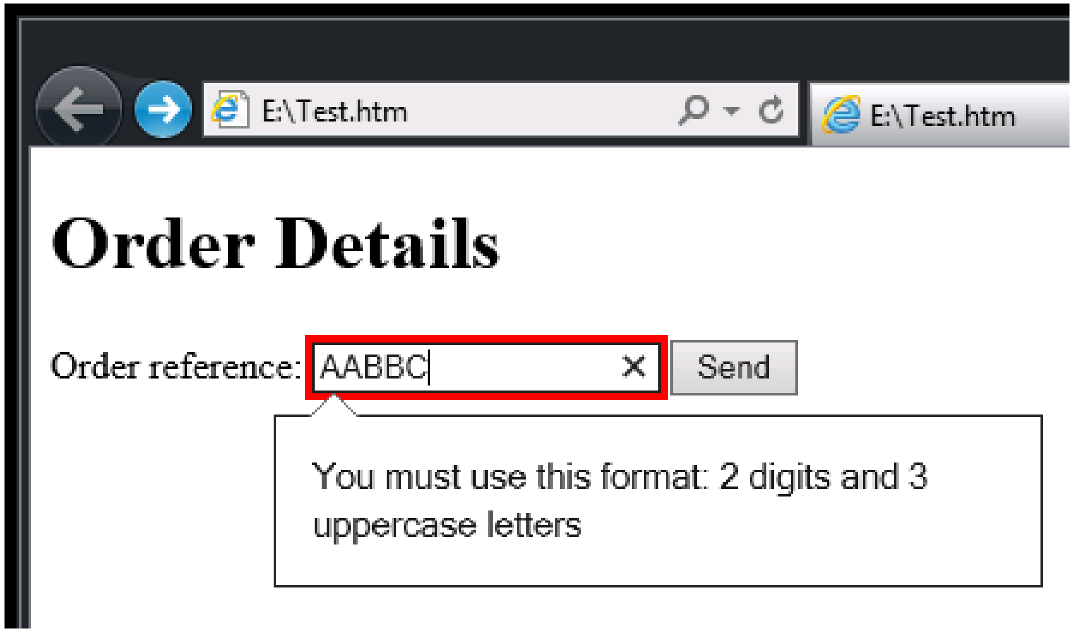
Enter a number between 0 and 100: 101

You must enter a value between 0 and 100

Validating Text Input

- Use the **pattern** attribute to validate text-based input by using a regular expression

```
<input id="orderRef" name="orderReference" type="text"
  pattern="[0-9]{2}[A-Z]{3}" title="2 digits and 3 uppercase letters" />
```



Order Details

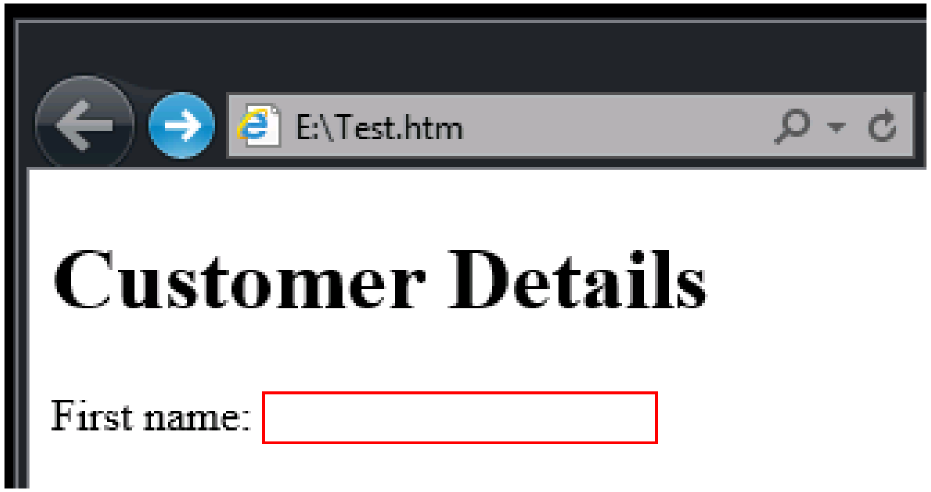
Order reference: AABBC

You must use this format: 2 digits and 3 uppercase letters

Styling Fields to Provide Feedback

- Use CSS to style input fields
  - Use the **valid** and **invalid** pseudo-classes to detect fields that have passed or failed validation

```
input {
  border: solid 1px;
}
input:invalid {
  border-color: #f00;
}
input:valid {
  border-color: #0f0;
}
```



Customer Details

First name:

Lesson 3: Validating User Input by Using JavaScript

- Handling Input Events
- Validating Input
- Ensuring that Fields are Not Empty
- Providing Feedback to the User
- Demonstration: Creating a Form and Validating User Input

Handling Input Events

- Catch the **submit** event to validate an entire form
  - Return true if the data is valid, false otherwise
  - The form is only submitted if the **submit** event handler returns true
- Catch the **input** event to validate individual fields on a character-by-character basis
  - If the data is not valid, display an error message by using the **setCustomValidity** function
  - If the data is valid, reset the error message to an empty string