Experiment No. 02

Title: Design Website forms using HTML5.

Batch: B-4 **Roll No.: 16010422234** **Name: Chandana Ramesh Galgali**

Experiment No.: 2

Aim: To design website forms to accept data from the user through the HTML 5.0 form elements.

Resources needed: HTML 5.0 editor

Theory:

Basics of HTML Forms:

HTML forms contain **form elements**. Form elements are different types of input elements, checkboxes, radio buttons, submit buttons, and more.
For Example:
<input type="text"> defines a one-line input field for **text input.**
<input type="radio"> defines a **radio button**.

The other input elements are:
- Checkboxes
- Button
- Textarea
- Select

The different attributes of forms are:

The Action Attribute: The **action attribute** defines the action to be performed when the form is submitted. The common way to submit a form to a server is by using a submit button. Normally, the form is submitted to a web page on a web server.
For example:
<formaction="action_page.php">

The Method Attribute: The method attribute **specifies the HTTP method (GET or POST) to be used when submitting the forms.**
For example:
<formaction="action_page.php"method="get"> or
<formaction="action_page.php"method="post">

A history of HTML5 forms:

The forms section of HTML5 was originally a specification titled Web Forms 2.0 that added new types of controls for forms. Started by Opera and edited by then-Opera employee Ian Hickson, it was submitted to the W3C in early 2005. The work was initially carried out under the W3C. It was then combined with the Web Applications 1.0 specification to create the basis of the breakaway Web Hypertext Application Technology Working Group (WHATWG) HTML5 specification.

Using HTML5 design principles

One of the best things about HTML5 forms is that you can use almost all of these new input types and attributes right now. They don't even need any shivs, hacks, or workarounds. That isn't to say they're all "supported" right now, but they do cool things in modern browsers that do support them-and degrade gracefully in browsers that don't understand them. This is thanks to HTML5's design principles. In this instance we're specifically referring to the principle of graceful degradation. In essence, this means that there's no excuse for not using these features right now. In fact, it means you're ahead of the curve.

HTML5 form attributes

There are 14 new attributes provided by HTML5

placeholder	autofocus
autocomplete	required
pattern	list
multiple	novalidate
formnovalidate	form
formaction	formenctype
formmethod	formtarget

1. placeholder

First up is the placeholder attribute, which allows us to set placeholder text as we would currently do in HTML4 with the value attribute. It should only be used for short descriptions. For anything longer, use the title attribute. The difference from HTML4 is that the text is only displayed when the field is empty and hasn't received focus. Once the field receives focus (e.g., you click or tab to the field), and you begin to type, the text simply disappears. It's very similar to the search box you see in Safari (see Figure 1).

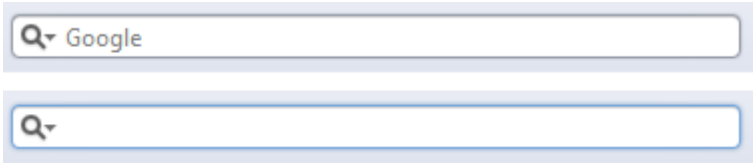


Figure 1. Browser search box in Safari without and with focus Let's have a look at how to implement the placeholder attribute.

<inputtype="text"name="user-name"id="user-name"placeholder="atleast 3 characters">

Figure 2 shows the placeholder attribute working in Chrome.

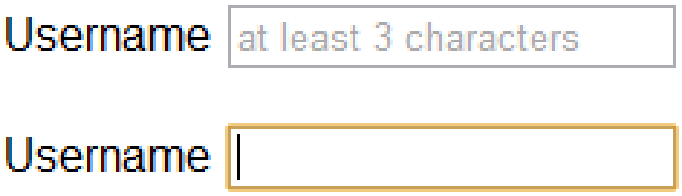


Figure 2. Placeholder attribute support in Chrome, unfocused and focused

2. autofocus

autofocus does exactly what it says on the tin. Adding it to an input automatically focuses that field when the page is rendered. It is a Boolean attribute (except if you are writing XHTML5; see the note) and is implemented as follows:
<input type="text"name="first-name"id="first-name" autofocus>

3. autocomplete

The autocomplete attribute helps users complete forms based on earlier input. The default state is set to on. This means that generally we won't have to use it. However, if you want to insist that a form field be entered each time a form is completed (as opposed to the browser auto filling the field), you would implement it like so:
<input type="text"name="tracking-code"id="tracking-code"autocomplete="off">
The autocomplete state on a field overrides any autocomplete state set on the containing form element.

4. required

The required attribute doesn't need much introduction; like autofocus, it does exactly what you'd expect. By adding it to a form field, the browser requires the user to enter data into that field before submitting the form. required is a Boolean attribute, like autofocus. Let's see it in action.
<input type="text"id="given-name"name="given-name" required>

New Input Types in HTML5

- color
- date
- datetime
- datetime-local
- email
- month
- number
- range
- search
- tel
- time
- url
- week

The new Elements added by HTML5

list and the datalist element

The list attribute enables the user to associate a list of options with a particular field. The value of the list attribute must be the same as the ID of a datalist element that resides in the

same document. The following example shows how list and datalist are combined (see Figure)

```
<label>Your favorite fruit:
<datalist id="fruits">
<option value="Blackberry">Blackberry</option>
<option value="Blackcurrant">Blackcurrant</option>
<option value="Blueberry">Blueberry</option>
<!-- … -->
</datalist>
If other, please specify:
<input type="text" name="fruit" list="fruits">
</label>
```

By adding a select element inside the datalist you can provide superior graceful degradation than by simply using an option element.

```
<label>Your favorite fruit:
<datalist id="fruits">
<select name="fruits">
<option value="Blackberry">Blackberry</option>
<option value="Blackcurrant">Blackcurrant</option>
<option value="Blueberry">Blueberry</option>
<!-- … -->
</select>
If other, please specify:
</datalist>
<input type="text" name="fruit" list="fruits">
</label>
```

Browser support for list and datalist is currently limited to Opera 9.5+ (see Figure 5), Chrome 20+, Internet Explorer 10 and Firefox 4+.

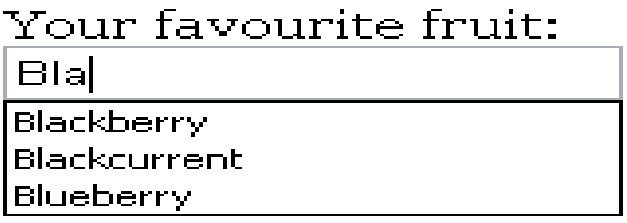


Figure 3: The datalist element rendered in Opera

Attributes of the Form tag:

- Formaction
- Formenctype
- Formmethod
- Formtarget
- Novalidate
- formnovalidate

The novalidate and formnovalidate attributes indicate that the form shouldn't be validated when submitted. They are both Boolean attributes. formnovalidate can be applied to submit or image input types. The novalidate attribute can be set only on the form element.
The following example shows how to use formnovalidate:

<form action="process.php">
<label for="email">Email:</label>
<input type="text" name="email"value="gordo@example.com">
<input type="submit" formnovalidate value="Submit">
</form>
And this example shows how to use novalidate:
<form action="process.php" novalidate>
<label for="email">Email:</label>
<input type="text" name="email"value="gordo@example.com">
<input type="submit" value="Submit">
</form>

Activity:

Design a form (eg. Registration form/feedback form/admission form etc) with HTML 5.0 new form features.

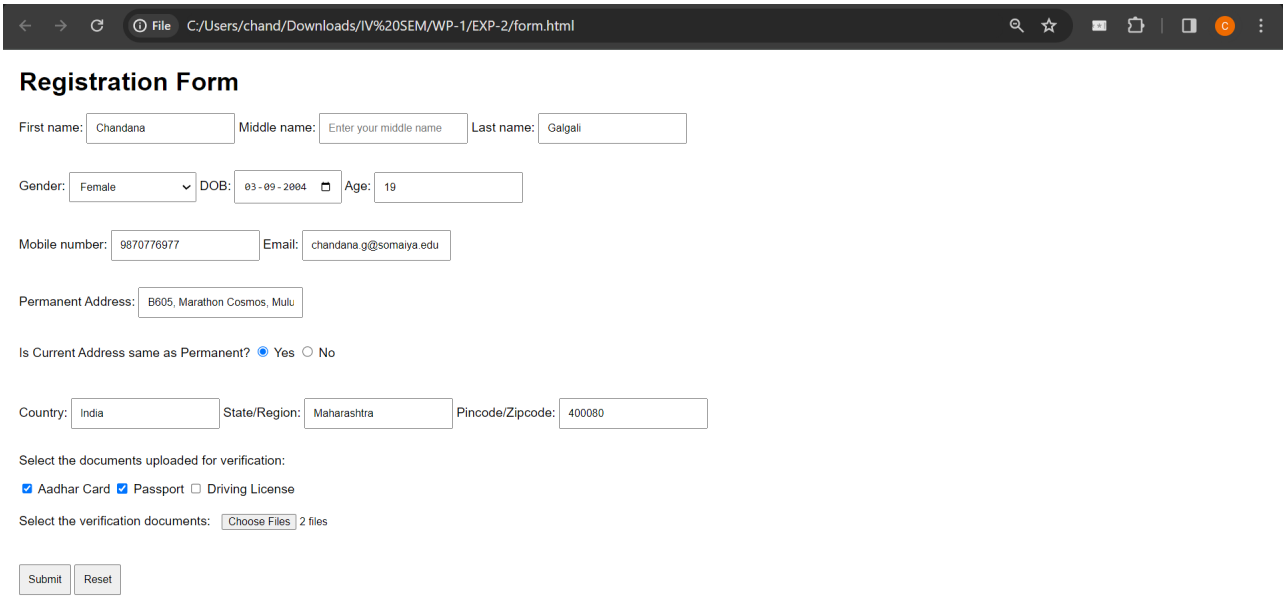
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <style>
      body {
            font-family: Arial, sans-serif;
            margin: 20px;
        }
        input, select {
            width: auto;
            padding: 10px;
            margin-bottom: 15px;
        }
        .address-fields {
            justify-content: space-between;
        }
        .document-upload {
            display: none;
        }
    </style>
</head>
<body>
    <h1>Registration Form</h1>
    <form action="/backend.php">
        <label for="fname">First name:</label>
        <input type="text" id="fname" name="fname" required placeholder="Enter
your first name">
        <label for="mname">Middle name:</label>
        <input type="text" id="mname" name="mname" placeholder="Enter your
middle name">
```

```
        <label for="lname">Last name:</label>
        <input type="text" id="lname" name="lname" required placeholder="Enter
your last name">
        <br><br>
        <label for="gender">Gender:</label>
        <select id="gender" name="gender" required>
            <option value="" disabled selected>Select your gender</option>
            <option value="male">Male</option>
            <option value="female">Female</option>
            <option value="other">Other</option>
        </select>
        <label for="dob">DOB:</label>
        <input type="date" id="dob" name="dob" required
onchange="calculateAge()">
        <label for="age">Age:</label>
        <input type="text" id="age" name="age" readonly><br><br>
        <label for="phone">Mobile number:</label>
        <input type="tel" id="phone" name="phone" required
placeholder="9870776977" pattern="[1-9]{1}-[0-9]{9}">
        <label for="email">Email:</label>
        <input type="email" id="email" name="email" required placeholder="Enter
your email">
        <br><br>
        <div class="address-fields">
            <div>
                <label for="permanent-address">Permanent Address:</label>
                <input type="text" id="permanent-address"
name="permanent-address" required placeholder="Enter your permanent address"
size="23">
            </div><br>
            <div>
                <label>Is Current Address same as Permanent?</label>
                <label><input type="radio" name="same-address" value="yes"
onclick="toggleCurrentAddress(true)"> Yes</label>
                <label><input type="radio" name="same-address" value="no"
onclick="toggleCurrentAddress(false)"> No</label>
            </div>
        </div><br>
        <div id="current-address-section" style="display:none;">
            <label for="current-address">Current Address:</label>
            <input type="text" id="current-address" name="current-address"
placeholder="Enter your current address">
        </div><br>
        <label for="country">Country:</label>
        <input type="text" id="country" name="country" required
placeholder="Country">
```

```
        <label for="state">State/Region:</label>
        <input type="text" id="state" name="state" required
placeholder="State/Region">
        <label for="pincode">Pincode/Zipcode:</label>
        <input type="text" id="pincode" name="pincode" pattern="\d{6}" required
placeholder="Enter your pincode/zipcode"><br>
        <p>Select the documents uploaded for verification:</p>
        <input type="checkbox" id="aadharcard" name="aadharcard" value="Aadhar
Card">
        <label for="aadharcard"> Aadhar Card</label>
        <input type="checkbox" id="passport" name="passport" value="Passport">
        <label for="passport"> Passport</label>
        <input type="checkbox" id="drivinglicense" name="drivinglicense"
value="Driving License">
        <label for="drivinglicense"> Driving License</label><br>
        <div class="document-upload" id="document-upload-section">
            <label for="myfile">Select the verification documents:</label>
            <input type="file" id="myfile" name="myfile" multiple><br><br>
        </div>
        <input type="submit" onclick="alert('Submitting the response!')"
value="Submit">
        <input type="reset" value="Reset">
        <script>
          function calculateAge() {
            var dobInput = document.getElementById("dob");
            var ageInput = document.getElementById("age");
            if (dobInput.value) {
              var dob = new Date(dobInput.value);
              var currentDate = new Date();
              var age = currentDate.getFullYear() - dob.getFullYear();
              if (currentDate.getMonth() < dob.getMonth() ||
(currentDate.getMonth() === dob.getMonth() && currentDate.getDate() <
dob.getDate())) {
                  age--;
              }
              ageInput.value = age;
            } else {
                ageInput.value = "";
             }
            }
        </script>
        <script>
            function toggleCurrentAddress(isSame) {
                var currentAddressSection =
document.getElementById("current-address-section");
                currentAddressSection.style.display = isSame ? "none" :
```

```
"block";
            }
        </script>
        <script>
          function toggleDocumentUpload() {
              var documentUploadSection =
document.getElementById("document-upload-section");
              var aadharCheckbox = document.getElementById("aadharcard");
              var passportCheckbox = document.getElementById("passport");
              var drivingLicenseCheckbox =
document.getElementById("drivinglicense");
              if (aadharCheckbox.checked || passportCheckbox.checked ||
drivingLicenseCheckbox.checked) {
                  documentUploadSection.style.display = "block";
              } else {
                  documentUploadSection.style.display = "none";
              }
          }
          document.getElementById("aadharcard").addEventListener("click",
toggleDocumentUpload);
          document.getElementById("passport").addEventListener("click",
toggleDocumentUpload);
          document.getElementById("drivinglicense").addEventListener("click",
toggleDocumentUpload);
      </script>
    </form>
</body>
</html>
```

Results: (Program printout with output / Document printout as per the format)



Questions:

1. What is the use of multiple in list and datalist elements?

<select> Element: In HTML, the <select> element is used to create a drop-down list. The multiple attribute, when added to a <select> element, allows users to select multiple options from the list instead of just one. This is useful when you want to enable users to choose multiple items from a set of options.

Example:
```
<select multiple>
   <option value="option1">Option 1</option>
   <option value="option2">Option 2</option>
   <option value="option3">Option 3</option>
</select>
```

<datalist> Element: The <datalist> element is used in conjunction with an <input> element to provide a predefined list of options. The multiple attribute in the <input> element allows users to select multiple values from the provided list.

Example:
```
<input list="options" multiple>
<datalist id="options">
   <option value="option1">
   <option value="option2">
   <option value="option3">
</datalist>
```

2. What is the importance of pattern attributes?

The pattern attribute is used with the <input> element, particularly with input types like text, tel, email, url, and password. It specifies a regular expression pattern that the input's value must match to be considered valid. This is valuable for enforcing specific formats for user inputs. For example, you can use the pattern attribute to ensure that a user enters a valid email address or follows a particular password format.

Example:
```
<input type="text" pattern="[0-9]{3}-[0-9]{2}-[0-9]{4}" title="Enter a valid SSN">
```

3. What are the three types of button that can be used in form?

Submit Button (<input type="submit">): Used to submit the form data to the server. Example:
```
<input type="submit" value="Submit">
```

Reset Button (<input type="reset">): Resets all form fields to their default values. Example:
```
<input type="reset" value="Reset">
```

Button (<button type="button">): Used for custom JavaScript actions within the form. Example:
```
<button type="button" onclick="myFunction()">Click Me</button>
```

Outcomes: Create Web pages using HTML 5 and CSS

Conclusion: (Conclusion to be based on the outcomes achieved)
The experiment demonstrates that HTML 5.0 form elements provide a powerful and flexible foundation

for designing effective data collection mechanisms on websites. By leveraging these elements appropriately, web developers can enhance the user experience, improve data integrity, and streamline the process of gathering information from users.

Grade: AA / AB / BB / BC / CC / CD /DD

Signature of faculty in-charge with date:

References:

Books/ Journals/ Websites:

- "HTML5: Black Book", Dreamtech Publication.
- "Web Technologies: Black Book", Dreamtech Publication.
- http://www.w3schools.com