Tutorial No. 03

Title: HTML5 Form handling.

Batch: B2 **Roll No.: 1914078** **Tutorial No.: 3**

Aim: HTML5 Form handling.

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:
<form**action="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 <form action="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.

<input type="text"name="user-name"id="user-name"placeholder="at least 3 characters">

That's it! We can hear you thinking, "What's so great about that? I've been doing it with JavaScript for years." Yes, that's true. However, with HTML5, it's part of the browser, meaning less scripting is required for a more accessible, cross-browser solution (even when JavaScript is disabled). Figure 2 shows the placeholder attribute working in Chrome.

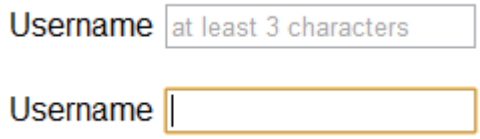


Figure 2. Placeholder attribute support in Chrome, unfocused and focused

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 autofilling 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; likeautofocus, 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+.

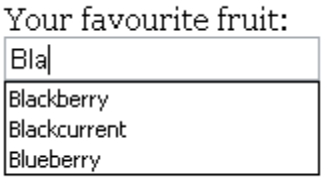


Figure :Thedatalist element rendered in Opera

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 and test all the new form features(elements, attributes ,input types)on random information.

Results: Program uploaded on LMS/MS Team /Google Classroom Assignment created along with output screenshot in write-up)

Code (with very little css) :-

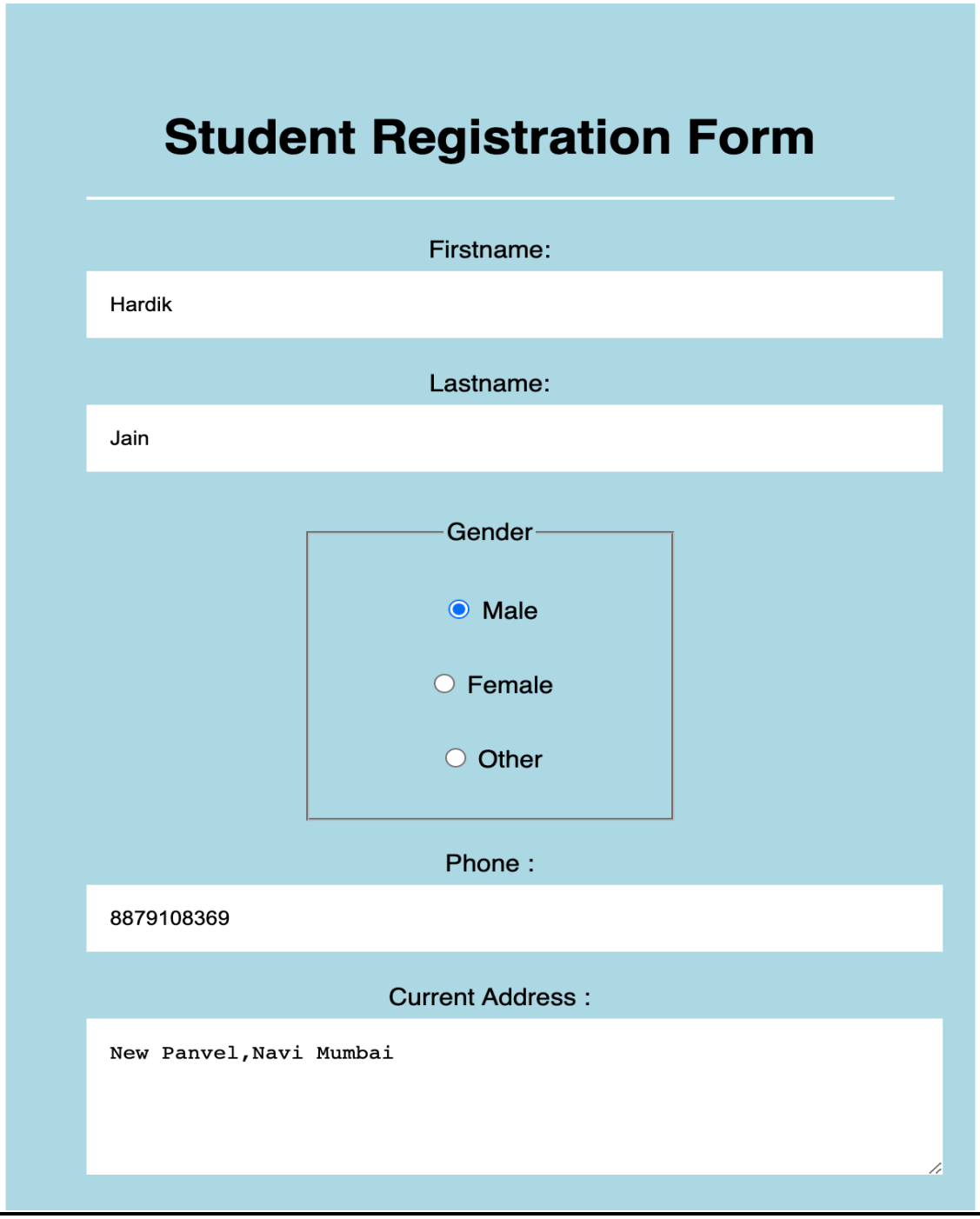
```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1">
<!--Here , we have used CSS and added various alignment and enhancement features. -->
<style>
body{
  font-family: sans-serif;
}

.container {
  padding: 50px;
  background-color: lightblue;
  text-align:center;
  width:500px;
  margin: auto;
}

input[type=text], input[type=password], input[type=email], textarea {
  width: 100%;
  padding: 15px;
  margin: 5px 0 22px 0;
  display: inline-block;
  border: none;
  background: white;
}
input[type=text]:focus, input[type=password]:focus ,input[type=email]:focus {
  outline: none;
}
div {
       padding: 10px 0;
       margin-bottom:10px;
     }
hr {
  border: 1px solid white;
  margin-bottom: 25px;
}
fieldset {
           margin: auto;
           width: 200px;
       }
</style>
</head>
<body>
<form action="" target="_self" method="POST" novalidate>
  <div class="container">
  <center>  <h1> Student Registeration Form</h1> </center>
  <hr>
```

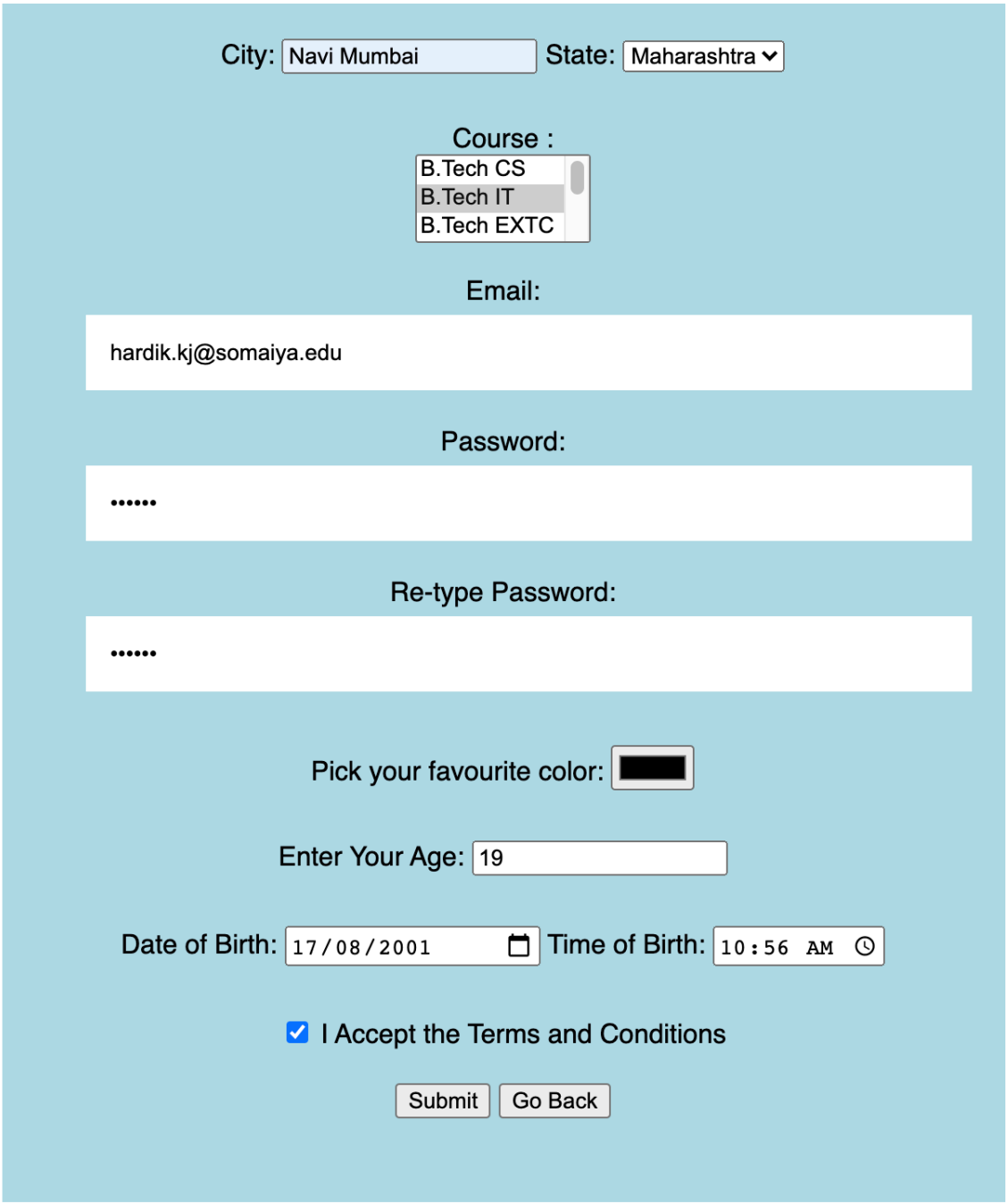
```
  <label> Firstname: </label>
<!--Here , we have used HTML Placeholder feature and input type text for entering the first and last name -->

<input type="text" name="firstname" placeholder= "Firstname" size="15" required />
<label> Lastname: </label>
<input type="text" name="lastname" placeholder="Lastname" size="15" required />
<!--Here , we have used HTML feildset feature and input type radio to add radio button feature to the gender feature. -->

<div>
            <fieldset>
               <legend>Gender</legend>
               <br>
               <div>
                  <input type="radio" name="gender" id="male" value="Male">
                  <label class="form-check-label" for="male">Male</label>
               </div>
               <div>
                  <input type="radio" name="gender" id="female" value="Female">
                  <label class="form-check-label" for="female">Female</label>
               </div>
                <div>
                  <input type="radio" name="gender" id="Other" value="Other">
                  <label class="form-check-label" for="Other">Other</label>
               </div>

            </fieldset>
 </div>
<label>
Phone :
</label>
<!--Here, we have used HTML input type text for entering phone number we used the size feature to limit the  text entered to 10. -->

<input type="text" name="phone" placeholder="Phone no." size="10"/ required>

<label>
Current Address :
</label>
<!--Here , we have used HTML Textarea feature to enter the address -->
<textarea cols="80" rows="5" placeholder="Current Address" value="address" required>
</textarea>
<div>
            <label for="city">City:
            <input list="city" id="city" placeholder="Type to search...">
        <!--Here , we have used HTML datalist feature. By adding a select element inside the datalist we can provide superior graceful degradation than by simply using an option element -->
            <datalist id="city">
```

```
                                    <select name="city">
                <option value="Mumbai">Mumbai</option>
                <option value="Pune">Pune</option>
                <option value="Delhi">Delhi</option>
                                            <option value="Bengaluru">Bengaluru</option>
                <option value="Navi Mumbai">Navi Mumbai</option>
                <option value="Thane">Thane</option>
              </select>
              </datalist>
              </label>

          <!--Here , we have used HTML dropdown feature to select state .-->
              <label for="state">State:</label>
              <select id="day">
                <option selected>State</option>
                <option value="Maharashtra">Maharashtra</option>
                <option value="Madhya Pradesh">Karnataka</option>
                <option value="Delhi">Delhi</option>
              </select>

</div>
<div>
<label for="courses">
Course :
</label>
<br>
        <!--Here , we have used HTML Multiple feature to select the course name-->

<select id="courses" name="courses" size="3" multiple>

<option value="B.Tech CS">B.Tech CS</option>
<option value="B.Tech IT">B.Tech IT</option>
<option value="B.Tech EXTC">B.Tech EXTC</option>
<option value="B.Tech MECH">B.Tech MECH</option>
<option value="B.Tech ETRX">B.Tech ETRX</option>
<option value="M.Tech">M.Tech</option>
</select>
</div>
        <!--Here , we have used HTML E-mail and Password input types to accept email and
password from the user-->

 <label for="email">Email:</label>
 <input type="email" placeholder="Enter Email" name="email" required>

   <label for="psw">Password:</label>
   <input type="password" placeholder="Enter Password" name="psw" required>

   <label for="psw-repeat">Re-type Password:</label>
   <input type="password" placeholder="Retype Password" name="psw-repeat" required>
            <div>
```

```
  <!--Here , we have used HTML Input types color, number date and time to accept these from the user

        <label for="color">Pick your favourite color:</label>
            <input type="color" id="color">
          </div>
          <div>
            <label for="num">Enter Your Age:</label>
            <input type="number" id="num" required>
          </div>

          <div>
            <label for="dob">Date of Birth:</label>
            <input type="date" id="dob" required>

            <label for="t">Time of Birth:</label>
            <input type="time" id="t" required>
          </div>
<div>
      <!--Here , we have used HTML Input type checkbox-->

          <input type="checkbox" id="exampleCheck1">
          <label for="exampleCheck1">I Accept the Terms and Conditions</label>
</div>
      <!--Here , we have used HTML Input type submit to add submit button and also added a go back button.-->

        <input type="submit" value="Submit" formnovalidate>
        <button>Go Back</button>
        <br>

</form>
</body>
</html>
```

Output:-


Student Registration Form
Firstname:
Hardik
Lastname:
Jain
Gender
Male
Female
Other
Phone :
8879108369
Current Address :
New Panvel,Navi Mumbai



Questions:

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

Answer:

The multiple attribute is used to notate that multiple values should be able to be selected. The specification for the multiple attribute shows an example of usage with dataists. However, browsers have implemented the multiple attribute only for input elements where type=email, the type of input element shown in the specifications example. It would seem like the multiple attribute should also be followed for when type is not email,

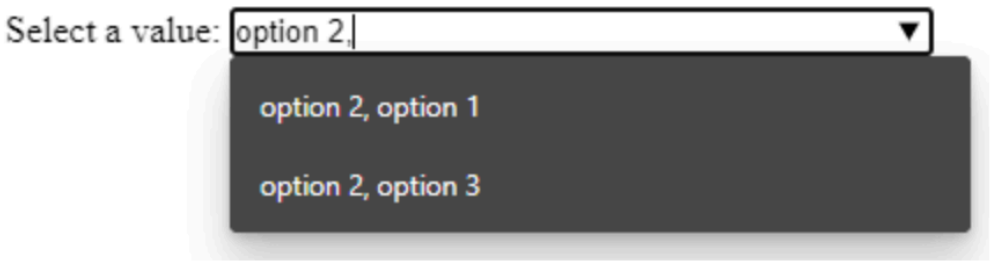
especially when type=text, however the specification does not make this clear, creating a large discrepancy between what people would expect, and what happens. This discrepancy also has the potential to lead to bad practices where type=email is used for non-email inputs.

Example:

The idea is to dynamically edit the option element of the datalist while you are typing to remove selected values and prefix all others with the already selected values. For instance, if the input value is option 2,, the HTML will be:

```
<input type="text" list="Suggestions" multiple value="option 2," />

<datalist id="Suggestions">
  <option>option 2, option 1</option>
  <option>option 2, option 3</option>
</datalist>
```

Output:



2. What is the importance of pattern attribute?
Answer:

Pattern attribute is used to specify the regular expression on which the input element value is checked. This attribute works with the following input types: text, password, date, search, email, etc. Use the Global title attribute to describe the pattern for helping the user

Syntax:

<input pattern = "regular_exp">

For example, say we have a username input in our form. There isn't a standard type for username, hence we use the regular text input type

```
<form action="somefile.php">

  <input type="text" name="username" placeholder="Username">
```

</form>

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

There are three types of buttons:

- submit — Submits the current form data. (This is default.)
- reset — Resets data in the current form.
- button — Just a button. Its effects must be controlled by something else.

It used to be the case that buttons primarily appeared in the context of forms, where the default submit behavior made perfect sense. Today, it is common to see buttons in all sorts of in-browser app contexts, and so it isn't always clear at first glance at the markup. Therefore, it is a good practice to *always* declare the type of a button explicitly.

Outcomes:

CO2: Create Web pages using HTML 5 and CSS **.**

Conclusion: (Conclusion to be based on the outcomes achieved)

Through this experiment we learnt HTML5 Form handling by designing a Registration form with HTML 5.0 and test all the new form features(elements, attributes ,input types)on random information. We have successfully written the HTML code for the registration form covering all the elements, attribute sand input types.

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