SVKM's NMIMS
Mukesh Patel School of Technology Management & Engineering (Mumbai Campus)
Computer Engineering Department (B.Tech Integrated Sem V)
Fundamentals of Website Designing
Lab Manual

PART A

(Part A: TO BE REFFERED BY STUDENTS)

Experiment No. 07

A.1 AIM:

Validation of HTML forms using JavaScript

A.2 Pre requisite:

Basic Knowledge of HTML and JavaScript

A.3 Outcome:

After successful completion of this experiment students will be able to:

1. Apply appropriate validations on the HTML forms using JavaScript

A.4 Theory:

JavaScript is the programming language of the Web. All modern HTML pages are using JavaScript.

JavaScript is one of **3** languages all web developers **MUST** learn:

1. **HTML** to define the content of web pages
2. **CSS** to specify the layout of web pages
3. **JavaScript** to program the behavior of web pages

JavaScript Form Validation

JavaScript can be used to validate data in HTML forms before sending off the content to a server.

Form data that typically are checked by a JavaScript could be:

- has the user left required fields empty?
- has the user entered a valid e-mail address?
- has the user entered a valid date?
- has the user entered text in a numeric field?

Assigning Names to Form Fields

In order to access your form fields in code, you need to assign names to the form and each of the fields. You do this by using the Name attribute.

In the code provided previously you can find the following line.

```
<p>Name: <input type="text" size="65"></p>
```

When you add the name attribute, the code looks like the following:

```
<p>Name: <input type="text" size="65" name="Name"></p>
```

Once you have assigned names for your form and all of the form elements, your form code should resemble the following code.

```
<form method="post" action="mailto:Frank@cohowinery.com" name="ContactForm">
   <p>Name: <input type="text" size="65" name="Name"></p>
   <p>E-mail Address:  <input type="text" size="65" name="Email"></p>
   <p>Telephone: <input type="text" size="65" name="Telephone"><br>
      <input type="checkbox" name="DoNotCall"> Please do not call me.</p>
   <p>What can we help you with?
      <select type="text" value="" name="Subject">
         <option>  </option>
         <option>Customer Service</option>
         <option>Question</option>
         <option>Comment</option>
         <option>Consultation</option>
         <option>Other</option>
      </select></p>
   <p>Comments:  <textarea cols="55" name="Comment">  </textarea></p>
   <p><input type="submit" value="Send" name="submit"><input type="reset" value="Reset"
name="reset"></p>
</form>
```

Writing the Validation Script

Creating the Function

```
<script>
function ValidateContactForm()
{

}
</script>
```

Creating Field Variables

To access the form fields within the code, you should create variables.
Variables allow you to temporarily store values. Variables are not required, but they make accessing each field easier. For example, without a variable, you would have to type document.ContactForm.Name every time you needed to access the Name field. However, after you assign a variable to the field, you can use the variable to access the field. Variables require less typing and help make your code more readable.
The contact form code in this article contains six fields to which you assigned name attribute values; therefore, you need six variables, one for each field.

```
function ValidateContactForm()
{
   var name = document.ContactForm.Name;
   var email = document.ContactForm.Email;
   var phone = document.ContactForm.Telephone;
   var nocall = document.ContactForm.DoNotCall;
   var what = document.ContactForm.Subject;
   var comment = document.ContactForm.Comment;
}
```

Defining Required Fields

- You may want to require that users type something for some fields on your form. By default, when you first create a form, all fields are optional. Therefore, if you want to ensure that users fill in certain fields, you need to tell the form validation function to check each field by checking the value of the field.

```
if (name.value == "")
{
   window.alert("Please enter your name.");
   name.focus();
   return false;
}
```

If the user clicks the Submit button without entering a value in the Name field, the browser displays a message to remind the user to enter a name. The Insertion Point is placed in the Name field, and the user is returned to the form. The return statement with a value of false is necessary so that the rest of the code doesn't execute.

- You can add a similar if statement to verify that the user has selected an item from the drop-down list. In this case, you don't check the value of the field; you check the value of the selected item.
- The following code shows the validation for the What can we help you with drop-down list.

- Items in a drop-down list or list box begin numbering at zero, so if the index of the selected item is less than 0, the user hasn't selected anything.

```
if (what.selectedIndex < 1)
{
   alert("Please tell us how we can help you.");
   what.focus();
   return false;
}
```

Determining Whether Data Is Valid

- In some cases, you may want to verify that the data is valid based on a specified format. For example, you know that all e-mail addresses contain an at symbol (@) and at least one period (.).
- indexOf method to determine whether a string occurs within another string.
- For the e-mail address, you need to determine whether an @ symbol or a period occurs within the e-mail form field. If the value returned from the indexOf method is less than zero (or –1), the e-mail address is invalid, the validate function returns false, and focus is returned to the e-mail form field.

The following code shows validation for the Email field.

```
if (email.value == "")
{
   window.alert("Please enter a valid e-mail address.");
   email.focus();
   return false;
}

if (email.value.indexOf("@", 0) < 0)
{
   window.alert("Please enter a valid e-mail address.");
   email.focus();
   return false;
}

if (email.value.indexOf(".", 0) < 0)
{
   window.alert("Please enter a valid e-mail address.");
   email.focus();
   return false;
```

```
}
```

Multiple if statements have a cascading effect. If the first if statement indicates that the e-mail address contains a value, the second if statement runs, and if the second if statement indicates that the e-mail address contains an @ symbol, the third if statement runs. If any of the three if statements indicate an invalid e-mail address, a message is displayed to the user, the validation function returns false, and the user is returned to the form with the Insertion Point in the Email field.

Returning a Value from a Function

You can return a value from a custom function by using the return statement. As mentioned previously, each of the if statements shown previously contains a line that returns a Boolean value of false. When the validation function returns to the form, the return value tells the form whether to continue processing. A value of false tells the browser to stop processing the form. Therefore, at the end of the validation function, all form fields have you also need a statement to return a value of true. This means that validated and the form can continue processing.

Connecting the Form to the Script

After you write the validation function, you need to tell your form to run the script when someone clicks the Submit button. To do this, you need to use the onsubmit event for the form.

For more information, see onsubmit Event. (The onsubmit event occurs every time a form is submitted for processing, but it happens on the client side, so if all processing is done on the server, you don't need the onsubmit event.)
To cause your validation script to run every time a user submits the contact form, add the following code to the opening <form> tag.

```
onsubmit="return ValidateContactForm();"
```

Using an Event Handler Function

- Creating a function in response to an event is the same as creating any other function. Instead of adding the event script to the event in the opening tag of the element, you add the name of the function that handles the event.
- As shown in the previous section, you need the Telephone field disabled if the DoNotCall field is checked. But if the DoNotCall field is cleared, the Telephone field should be re-enabled. The following function shows how to do this using an if. . .else statement.

```
function EnableDisable(chkbx)
{
   if(chkbx.checked == true)
```

```
    {
      document.ContactForm.Telephone.disabled = true;
    }
    else
    {
      document.ContactForm.Telephone.disabled = false;
    }
}
```

- After you created the custom function, you can add the following code tag for the DoNotCall field.

```
onclick="EnableDisable(this);"
```

Notice that the EnableDisable function takes an argument. Therefore, when you call the event from within the element, you need to pass a value. In this case, you need to pass the current element, and the easiest way to do so is to use the this statement

A.5 Procedure/Task:

1. Complete the task given in previous experiment, i.e. exp04.

2. Insert the JavaScript validations in those forms as discussed and practiced in class *(All characters in a field validation, All Numbers in a field validation, Phone number validation, Password validation, Email id validation, Date format validation etc.)*

3. Prepare the document. Save and close the file and name it as **EXP06_Roll no_Batch no.**

SVKM's NMIMS

Mukesh Patel School of Technology Management & Engineering (Mumbai Campus)

Computer Engineering Department (B.Tech Integrated Sem V)

Fundamentals of Website Designing

Lab Manual

PART B

(PART B: TO BE COMPLETED BY STUDENTS)

(Students must submit the soft copy as per following segments within two hours of the practical. The soft copy must be uploaded on the Blackboard or emailed to the concerned lab in charge faculties at the end of the practical in case the there is no Black board access available)

Roll No. :	Name:
Class :	Batch :
Date of Experiment :	Date/Time of Submission :
Grade :	

B.1 Code:

(Paste your Code here)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Registration Form</title>
</head>
<body style="font-family: Arial, sans-ser        <label
for="password">Password:</label><br>
        <input type="password" id="password" placeholder="Enter your password"
style="width: 100%; padding: 10px; margin-bottom: 10px;"><br>

        <label for="con
```

B.2 Output

(Take screen shots of the output at run t

BL f**Q2:** ame" required>
  <input type="submit" value="Submit">
</f56-7890).

```
html
Copy code
<form>
  <label for="phone">Phone:</label>
  <input type="text" id="phone" name="phone" pattern="\d{3}-\d{3}-
\d{4}" required>
  <small>Format: 123-45
```

3. Type Validator This ensures that the input is of a specific type, such as an email address.

```
html
Copy code
<form>
  <label for="email">Email:</label>
  <input type="email" id="email" name="email" required>
  <input type="submit" value="Submit">
</form>
```

These examples demonstrate how to implement basic validation in HTML forms to ensure users provide valid and necessary information.