Chapter 9

Form Implementation and Validation

Learning Objectives

After completing this chapter, you will be able to:

- *Understand the HTML form elements*
- *Understand form methods and action*
- *Understand the concept of JavaScript*
- *Understand the $_GET and $_POST superglobal variables*
- *Understand regular expressions*

INTRODUCTION

HTML forms are very important part of any dynamic website. An HTML form plays a key role to make any website interactive and user friendly. The forms are generally used to get the user details. You can create a form for your website using HTML 5. The PHP language is used along with HTML 5 to create dynamic forms in the website and to validate these forms, JavaScript is used. In this chapter, you will learn about HTML 5 form elements, form methods and action, JavaScript, and so on in detail.

HTML FORM ELEMENTS

HTML form elements are used to create a form in a website. The HTML form elements are form tags used to create different elements of form such as labels, buttons, drop-down list, input controls, and so on. The syntax to create HTML form is as follow:

```
<form>
 <!-- Form elements -->
</form>
```

In the given syntax, **<form>** is the start tag of HTML form body and **</form>** is the end tag of HTML form body. The form elements such as input text-box, labels, textarea, and so on are placed in between **<form>** and **</form>** tags.

Note

Every content from the start tag to the end tag of HTML is known as HTML element.

Attributes

An attribute in HTML provides additional information to the HTML elements. It is always written inside HTML start tag (HTML element) and contains a value. The syntax of attribute is as follow:

```
<tagname attribute="value">
```

In the given syntax, **<tagname>** represents any start tag (element) of HTML. Here, **attribute** represents any element attribute which provides additional information to the element and **value** represents any value related to the attribute also known as attribute description.

It is not necessary that every attribute will have a value. In that case, the syntax for attribute will be as follow:

```
<tagname attribute>
```

Form Input Element

An HTML form can contain various input controls such as text-box, check-box, radio-buttons, and so on. Therefore, form input elements are required to create these form input controls.

Evaluation Copy. Do not reproduce. For information visit www.cadcim.com

The syntax of form input element is as follow:

```
<input type="value">
```

In the given syntax, **<input>** is the start tag (element) of the input control. The input element does not have end tag. Here, **type** is an input element attribute which provides additional information to the input element and **value** represents any value related to the **type** attribute.

There are various values available for the **type** attribute of the input element. Table 9-1 lists some of the possible values of the **type** attribute with their descriptions.

Table 9-1 *Values and descriptions of the* ***type*** *attribute*

Value	Description
text	It provides one-line input text box.
checkbox	It provides a single check box.
radio	It provides a single radio button.
submit	It provides a submit button for HTML form submission.
date	It provides the date calender for date selection.
email	It provides a input box with email validation in the form.
file	It provides the choose file button to browse a file to upload.
password	It provides the input box following the password pattern.
search	It provides the search box.
url	It provides the input box with URL validation in form.
button	It provides a button.

Example 1

The following program illustrates the use of HTML input elements. The program will create a static form containing various form controls by using the input element and display the output in the browser.

```
<!Doctype html>                                                    1
<html>                                                             2
<head>                                                             3
<title>Form</title>                                                4
</head>                                                            5
<body>                                                             6
<h1>Form</h1>                                                      7
   <form>                                                          8
         <p><b>Name:</b>                                           9
               <input type="text">                                10
         </p>                                                      11
         <p><b>Gender:</b><br>                                     12
               <input type="radio" name="gender" >Male             13
```

Evaluation Copy. Do not reproduce. For information visit www.cadcim.com

```
                <input type="radio" name="gender" >Female        14
            </p>                                                 15
            <p><b>Languages:</b><br>                             16
                <input type="checkbox"> PHP                      17
                <input type="checkbox"> HTML                     18
                <input type="checkbox"> java                     19
                <input type="checkbox"> c++                      20
            </p>                                                 21
            <p><b>Email:</b>                                     22
                <input type="email" required>                    23
            </p>                                                 24
            <p><b>Password:</b>                                  25
                <input type="password" required>                 26
            </p>                                                 27
            <p><b>URL:</b>                                       28
                <input type="url" required>                      29
            </p>                                                 30
            <p><b>Date:</b>                                      31
                <input type="date">                              32
            </p>                                                 33
            <p>                                                  34
                <input type="submit">                            35
            </p>                                                 36
            <p><b>File:</b>                                      37
                <input type="file">                              38
                <input type="button" value="upload">             39
            </p>                                                 40
        </form>                                                  41
    </body>                                                      42
    </html>                                                      43
```

Explanation

Line 8

<form>

In this line, **<form>** is the start tag of HTML form body.

Line 9 to Line 11

** Name:**
<input type="text">
** **

In these lines, ** ** and ** ** are the paragraph elements of HTML. **Name** is the normal HTML content written in between the paragraph elements and **<input type="text">** is the input element of HTML where **type** is the attribute that has value **text**. It will create a one-line input text box inside the form.

Line 12 to Line 15

** Gender:
**
<input type= "radio" name="gender" >Male

Evaluation Copy. Do not reproduce. For information visit www.cadcim.com

```
        <input type= "radio" name="gender" >Female
</p>
```

In these lines, **Gender:**, **Male**, and **Female** are the normal HTML content written in between the paragraph elements and **<input type="radio" name="gender">** is the input element of HTML where **type** is the attribute that has value **radio** and **name** is the attribute which can contain any name as its value. These lines will create two radio buttons in a paragraph inside the form.

Note

*Keep the values of attribute **name** same for the radio buttons of same category and keep the different value of attribute **name** for different category because if a user clicks on a radio button, it becomes checked and all other radio buttons with same name becomes unchecked.*

Line 16 to Line 21

```
<p><b>Languages:</b> <br>
                <input type="checkbox"> PHP
                <input type="checkbox"> HTML
                <input type="checkbox"> java
                <input type="checkbox"> c++
</p>
```

In these lines, **<input type="checkbox">** is the input element of HTML where **type** is the attribute that has value **checkbox**. These lines will create 4 check boxes in a paragraph inside the form.

Line 23

<input type="email" required>

In this line, **<input type="email" required>** is the input element of HTML where **type** is the attribute that has value **email** and **required** is the attribute of input element which makes the particular that has field mandatory to be filled. This lines will create a single field text box with email validations in a paragraph inside the form.

Line 26

<input type="password" required>

In this line, **<input type="password" required>** is the input element of HTML where **type** is the attribute that has value **password** and **required** is the attribute of input element which makes this particular input field mandatory to be filled. This lines will create a single field text box which will follow the password pattern inside the form.

Line 29

<input type="url" required>

In this line, **<input type="url" required>** is the input element of HTML where **type** is the attribute that has value **url** and **required** is the attribute of input element which makes this particular input field mandatory to be filled. This lines will create a single field text box with the URL validations in a paragraph inside the form.

Line 32

<input type="date">

In this line, **<input type="date">** is the input element of HTML where **type** is the attribute

Evaluation Copy. Do not reproduce. For information visit www.cadcim.com

that has value **date**. This lines will create a text box which will display the date calender for date selection in the drop-down menu inside the text box.

Line 35
<input type="submit">
In this line, **<input type="submit">** is the input element of HTML where **type** is the attribute that has value **submit**. This lines will create a submit button of the form.

Line 38
<input type="file">
In this line, **<input type="file">** is the input element of HTML where **type** is the attribute that has value **file**. This lines will create the **choose file** button to browse a file in the system.

Line 39
<input type="button" value="upload">
In this line, **<input type="button" value="upload">** is the input element of HTML where **type** is the attribute that has value **button** and **value** is the attribute of input element which provide label to the button. This line will create the button with the label **upload**.

Line 41
</form>
In this line, **</form>** is the end tag of HTML form body.

The output of Example 1 is displayed in Figure 9-1.

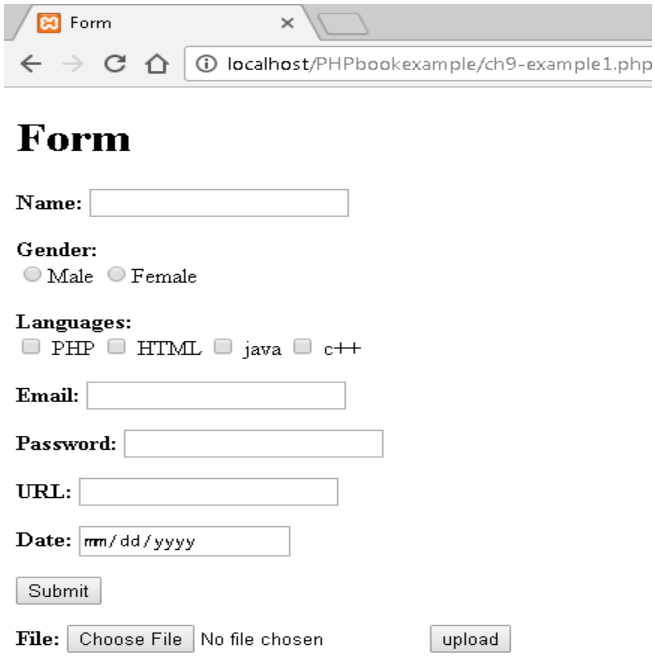

Figure 9-1 *The output of Example 1*

Evaluation Copy. Do not reproduce. For information visit www.cadcim.com

Form select Element

The form select element in HTML is used to create and define a drop-down list. The syntax of form select element is as follow:

```
<select name="val">
  <option value="item">Item</option>
  <option value="item2">Item2</option>
  ----
  ----
  <option value="itemN">ItemN</option>
</select>
```

In the given syntax, **<select>** and **</select>** are the select elements of HTML used to create drop-down list. Here, **name** is a select element attribute which provides additional information to the select element and **val** represents any name given to the select element through **name** attribute. To define the drop-down list, option element is used which is declared in between the select elements. Here, in the syntax, **<option>** and **</option>** are the option elements containing the attribute **value** which provides label or value to the option element and **Item** defined in between the option elements represents any name which is to be displayed in drop-down list. Therefore, to define N numbers of items in the drop-down list you need to define N numbers of option elements in between the select elements. By default the first item entered in the option elements is displayed in the drop-down text box.

Form datalist Element

The form datalist element in HTML is used to create a drop-down list which will display list of options when the user will start providing input data. The syntax of form datalist element is as follow:

```
<input list= "browsers">
  <datalist id="browsers">
    <option value="item1">
    <option value="item2">
    -----
    -----
    <option value="itemN">
  </datalist>
```

In the given syntax, **<datalist>** and **</datalist>** are the datalist elements of HTML used to create drop-down list which will display list of options when the user will start providing input data. Therefore, to get input from the user its important to have an input element to create the drop-down list using datalist elements. Here, **<input list="browsers">** is the input element that has an attribute **list** containing a value **browsers** which is fixed and cant not be changed. Now, the **<datalist>** element contains an attribute **id** having a value **browsers** which is also fixed and can not be changed. The attribute **id** provides identity to an element in an HTML. To define the drop-down list, option element is used which is declared in between the datalist elements. Here, in the syntax, **<option>** is the option element containing the attribute **value** which provides label or value to the option element which is to be displayed in drop-down list.

Evaluation Copy. Do not reproduce. For information visit www.cadcim.com

Therefore, to define N numbers of items in the drop-down list you need to define N numbers of option elements in between the datalist elements.

Note
The datalist element is not supported by Safari, IE9, and earlier versions of IE.

Form textarea Element

The form textarea element in HTML is used to create the multiple line input text box. The syntax of form textarea element is as follow:

```
<textarea>Content</textarea>
```

In the given syntax, **<textarea>** and **</textarea>** are the textarea elements of HTML used to create a multiple line input text box. Here, **Content** represents any content you want to write inside the text box which will get displayed in the browser. You can even skip writing content if you want user to provide the input.

Form button Element

The form button element in HTML is used to create the button. The syntax of form button element is as follow:

```
<button>button label</button>
```

In the given syntax, **<button>** and **</button>** are the button elements of HTML used to create the button. Here, **button label** represents any name or label of the button.

Form fieldset Element

The form fieldset element in HTML is used to create the form field (border) within which the form elements will exist. The syntax of form fieldset element is as follow:

```
<fieldset>
<!-- different form elements -->
</fieldset>
```

In the given syntax, **<fieldset>** and **</fieldset>** are the fieldset elements of HTML used to create the form field (border) within which the form elements will exist. You can place different form elements such as input, textarea, button, and so on elements in between fieldset elements.

Example 2

The following program will create a static form containing various form elements and display the output in the browser.

```
<!Doctype html>                                                    1
<html>                                                             2
<head>                                                             3
```

Evaluation Copy. Do not reproduce. For information visit www.cadcim.com

```
<title>Form</title>                                                  4
</head>                                                              5
<body>                                                               6
<h1>Form elements</h1>                                               7
   <form>                                                            8
         <fieldset style="width:300px">                              9
         <p><b>Name:</b>                                             10
               <input type="text">                                   11
         </p>                                                        12
         <p><b>Age group:</b>                                        13
               <input list="browsers">                               14
               <datalist id="browsers">                              15
                     <option value="10-19">                          16
                     <option value="20-29">                          17
                     <option value="30-39">                          18
                     <option value="40-49">                          19
                     <option value="50">                             20
               </datalist>                                           21
         </p>                                                        22
         <p><b>Language:</b>                                         23
               <select>                                              24
                     <option value="English">English</option>        25
                     <option value="Hindi">Hindi</option>            26
                     <option value="Spanish">Spanish</option>        27
                     <option value="French">French</option>          28
               </select>                                             29
         </p>                                                        30
         <p><b>Comment:</b>                                          31
               <textarea></textarea>                                 32
         </p>                                                        33
         <button type="submit">Send</button>                         34
         </fieldset>                                                 35
   </form>                                                           36
</body>                                                              37
</html>                                                              38
```

Explanation

Line 9

<fieldset style="width:300px">

In this line, **<fieldset>** is the start tag of HTML used to create the form field (border) within which the form elements will exist. Here, **style** is the attribute used to provide inline CSS to the fieldset element where **width** is CSS property followed by **:** (colon) and **300px** is its description. This will set the width of the fieldset to 300 pixels.

Line 14 to Line 21

<input list="browsers">

<datalist id="browsers">

<option value="10-19">

Evaluation Copy. Do not reproduce. For information visit www.cadcim.com

```
        <option value= "20-29">
        <option value="30-39">
        <option value="40-49">
        <option value="50">
</datalist>
```

In these lines, **<input list="browsers">** is the input element having an attribute **list** that contains a value **browsers** which is fixed and cannot be changed. The **<datalist>** and **</datalist>** are the datalist elements of HTML used to create drop-down list which will display list of options when the user will start providing input data. Here, the **<datalist>** element contains an attribute **id** having a value **browsers** which is also fixed and can not be changed. Next, **<option>** is the option element containing the attribute **value** which provides label or value to the option element which is to be displayed in drop-down list.

Line 24 to Line 29

```
<select>
        <option value="English">English</option>
        <option value="Hindi">Hindi</option>
        <option value="Spanish">Spanish</option>
        <option value="French">French</option>
</select>
```

In these lines, **<select>** and **</select>** are the select elements of HTML used to create drop-down list. Next, **<option>** and **</option>** are the option elements that contain the attribute **value** which provides label or value to the option element and **English**, **Hindi**, **Spanish**, and **French** defined in between the option elements are the items which is to be displayed in drop-down list.

Line 32

```
<textarea></textarea>
```

In this line, **<textarea>** and**</textarea>** are the textarea elements of HTML used to create a multiple line input text box.

Line 34

```
<button type="submit">Send</button>
```

In this line, **<button>** and **</button>** are the button elements of HTML used to create the button where **type** is an attribute of button element having value **submit**. Here, **Send** is the label given to the button.

Line 35

```
</fieldset>
```

In this line, **</fieldset>** is the end tag of HTML used to end the form field (border) within which the form elements will exist.

The output of Example 2 is displayed in Figure 9-2.

Evaluation Copy. Do not reproduce. For information visit www.cadcim.com

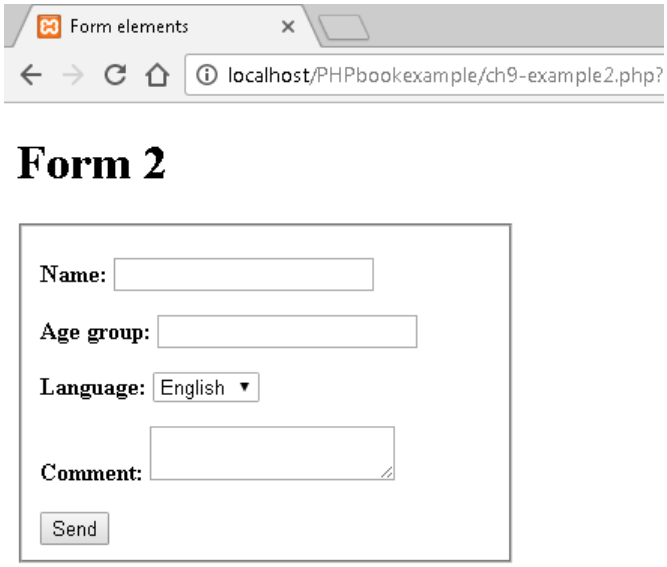

Figure 9-2 *The output of Example 2*

FORM METHODS AND ACTION

The form method specifies the method by which the form data will be sent. The form data is sent to the page which is specified by the form action. To specify the form method an attribute **method** is used inside the **<form>** tag and to specify the page where the data will be sent, the attribute **action** is used inside the **<form>** tag.

The attribute method can have two possible values that are as follow:

1. **get**
2. **post**

These form methods are discussed next.

The get Method

The **get** method is used when the user wants to send or append the form result into URL to bookmark the result. These results are appended into URL in the form of name/value pair. The **get** method is not secure as the input data of the form sent through the **get** method is visible to everyone in the URL. Therefore, never use the **get** method to send sensitive data such as password. The data that can be appended to URL through the **get** method is limited because the length of an URL is limited to 3000 characters approximately. You can use the **get** method to send data such as query strings similar to Google search.

The syntax of the **get** method is as follow:

```
<form method= "get">
```

In the given syntax, **<form>** is the start tag of HTML form where **method** is an attribute of form element and **get** is the method through which form data will be sent to the page.

Evaluation Copy. Do not reproduce. For information visit www.cadcim.com

Evaluation Copy. Do not reproduce. For information visit www.cadcim.com

The post Method

The **post** method is used to transfer data using HTTP header. This is the secure method to send the input data to another page, for example, to send sensitive data such as password you must use the **post** method instead of the **get** method. There is no data limit in post method, you can send multiple data to another page at a time. The data send by the **post** method can not be bookmarked and is not visible to everyone that is why it is more secure than the **get** method.

The syntax of the **post** method is as follow:

```
<form method= "post">
```

In the given syntax, **<form>** is the start tag of HTML form where **method** is an attribute of form element and **post** is the method through which form data will be sent to the page.

Form Action

The **action** is a form attribute used to specify the page where the form data will be sent by using the **get** or **post** method. The syntax of form action is as follow:

```
<form action="page.extension">
```

In the given syntax, **<form>** is the start tag of HTML form where **action** is an attribute of form element and **page.extension** represents the page where the form data will be sent.

RECEIVING FORM DATA IN PHP

The PHP file is used to receive or display the form data sent through the HTML file by using the two major form attributes: **method** and **action**. To receive the form data, you need to use some pre-defined PHP variables known as superglobals. You have already learned about superglobals in Chapter 4. In this section, you will learn and use some of them in PHP script to receive the data in PHP file as discussed next.

The $_GET Superglobal Variable

The **$_GET** superglobal variable is used in the PHP script when the form through which the PHP file is receiving the data is using the **get** method to send the data.

The syntax of using the **$_GET** superglobal variable to receive the form data in a PHP script is as follow:

```
$_GET["name"]
```

In the given syntax, **$_GET[]** is an array variable used to receive the data of form sent through the **get** method. Here, **name** represents the name of the form field through which the **$_GET[]** is receiving the data.

Example 3

In this example, you will create two programs, *ch9-example3-form.php* and *ch9-example3-formdata.php*, where the *ch9-example3-form.php* program will create a dynamic form containing various

form elements and send the form data to *ch9-example3-formdata.php* program file using the **get** method and the **action** attribute. The *ch9-example3-formdata.php* program file will receive the form data from *ch9-example3-form.php* program file using the **$_GET** superglobal variable to display the output in the browser.

The *ch9-example3-form.php* program file.

```
<!Doctype html>
<html>
<head>
<title>PHP Form</title>
</head>
<body>
<h1>PHP Form</h1>
   <form method="get" action="ch9-example3-formdata.php">
         <p><b>Name:</b>
               <input type="text" name="person" required>
         </p>
         <p><b>Gender:</b>
               <br>
               <input type="radio" name="gender">Male
               <input type="radio" name="gender">Female
         </p>
         <p><b>Email:</b>
               <input type="email" name="email" required>
         </p>
         <p><b>Password:</b>
               <input type="password" name="pass" required>
         </p>
         <input type="submit">
   </form>
</body>
</html>
```

The *ch9-example3-formdata.php* program file.

```
<!Doctype html>
<html>
<head>
<title>PHP Form Data</title>
</head>
<body>
<b>HELLO</b>
   <?php
echo $_GET["person"];
echo "<br>";
echo "<b>Your email address is: </b>";
echo $_GET["email"];
echo "<br>";
```

Evaluation Copy. Do not reproduce. For information visit www.cadcim.com

```
echo "<b>your password is: </b>". $_GET["pass"];                14
?>                                                                15
         <h3> Thank you for visiting :)</ h3>                     16
</body>                                                           17
</html>                                                           18
```

Explanation (ch9-example3-form.php)

Line 8

<form method="get" action="ch9-example3-formdata.php">

In this line, **<form>** is the start tag of HTML form body where **method** and **action** are the two attributes that has **get** and **ch9-example3-formdata.php** values, respectively.

Line 10

<input type="text" name="person" required>

In this line, **name** is the attribute that has value **person** which is the name given to the input text field.

Explanation (ch9-example3-formdata.php)

Line 9

echo $_GET["person"];

In this line, **$_GET []** is the superglobal variable used to receive the data of form sent through **get** method and **person** is the input text field name whose data is to be displayed.

The working of Line 12 and Line 14 is similar to Line 9

The output of Example 3 is displayed in Figure 9-3.

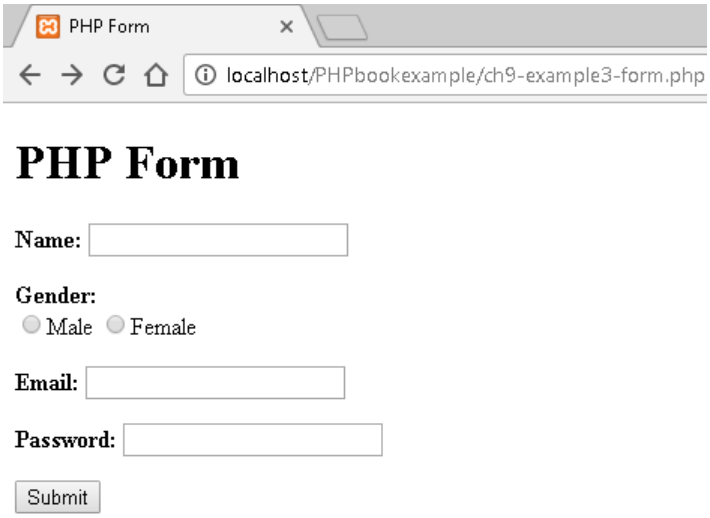

Figure 9-3 *The output of Example 3*

In Figure 9-3, the form is blank. Now, the user will enter the data to send it to another page as shown in Figure 9-4.

Evaluation Copy. Do not reproduce. For information visit www.cadcim.com

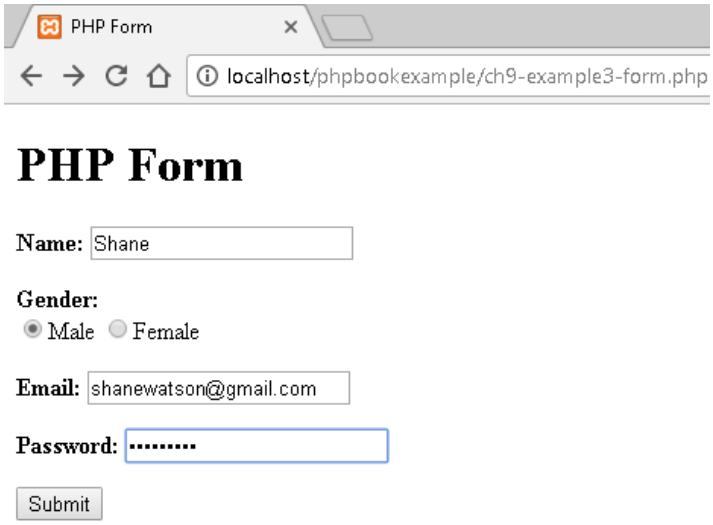

Figure 9-4 *The output of Example 3 with data in form*

In *ch9-example3-formdata.php* data is filled as shown in Figure 9-4. Now the user will submit this data and the data will be sent to *ch9-example3-formdata.php* program. This program will receive the data and display the data in the URL because **get** method is used to send the data and the output will also be displayed, as shown in Figure 9-5.

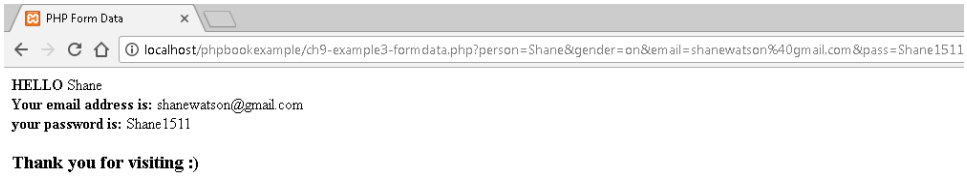

Figure 9-5 *The output of Example 3 displaying form data*

The $_POST Superglobla Variable

The **$_POST** superglobal variable is used in the PHP script when the form through which the PHP file is receiving the data is using the **post** method to send the data.

The syntax of using **$_POST** superglobal to receive the form data in a PHP script is as follow:

```
$_POST["name"]
```

In the given syntax, **$_POST[]** is an array variable used to receive the data of form sent through **post** method. Here, **name** represents the name of the form field through which the **$_POST[]** is receiving the data.

Example 4

In this example, you will create two programs,*ch9-example4-form.php* and *ch9-example4-formdata.php*, where the *ch9-example4-form.php* program will create a dynamic form containing various

Evaluation Copy. Do not reproduce. For information visit www.cadcim.com

form elements and send the form data to *ch9-example4-formdata.php* program file using the **post** method and the **action** attribute. The*ch9-example4-formdata.php* program file will receive the form data from *ch9-example4-form.php* program file using the **$_POST** superglobal variable to display the output in the browser.

The *ch9-example4-form.php* program file.

```
<!Doctype html>
<html>
<head>
<title>PHP post Form</title>
</head>
<body>
<h1>PHP Form 2</h1>
   <form method="post" action="ch9-example4-formdata.php">
         <p><b>Name:</b>
               <input type="text" name="person" required>
         </p>
         <p><b>Gender:</b>
               <br>
               <input type="radio" name="gender">Male
               <input type="radio" name="gender">Female
         </p>
         <p><b>Email:</b>
               <input type="email" name="email" required>
         </p>
         <p><b>Password:</b>
               <input type="password" name="pass" required>
         </p>
         <input type="submit">
   </form>
</body>
</html>
```

The *ch9-example4-formdata.php* program file.

```
<!Doctype html>
<html>
<head>
<title>PHP POST Form Data</title>
</head>
<body>
<b>Hello</b>
   <?php
echo $_POST["person"];
echo "<br>";
echo "<b>Your email address is: </b>";
echo $_POST["email"];
```

Evaluation Copy. Do not reproduce. For information visit www.cadcim.com

```
echo "<br>";                                                    13
echo "<b>your password is: </b>". $_POST["pass"];               14
?>                                                              15
        <h3> Thank you for visiting :)</ h3>                    16
</body>                                                         17
</html>                                                         18
```

Explanation (ch9-example4-form.php)

Line 8

<form method="post" action="ch9-example3-formdata.php">

In this line, **<form>** contain **method** and **action** attributes that has **post** and **ch9-example4-formdata.php** values, respectively.

Explanation (ch9-example4-formdata.php)

Line 9

echo $_POST["person"];

In this line, **$_POST []** is the superglobal variable used to receive the data of form sent through **post** method and **person** is the input text field name whose data is to be displayed.

The working of Line 12 and Line 14 is similar to Line 9

The output of Example 4 is displayed in Figure 9-6.

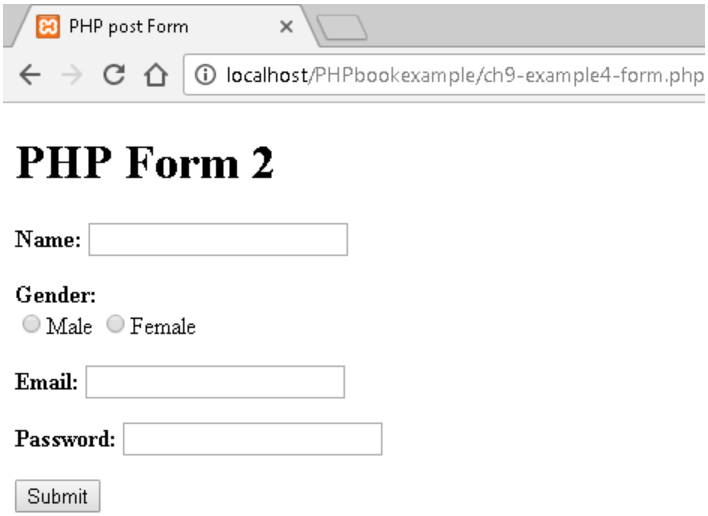

Figure 9-6 *The output of Example 4*

In Figure 9-6, the form is blank. Next, the user will enter the data to send it to another page, as shown in Figure 9-7.

Evaluation Copy. Do not reproduce. For information visit www.cadcim.com

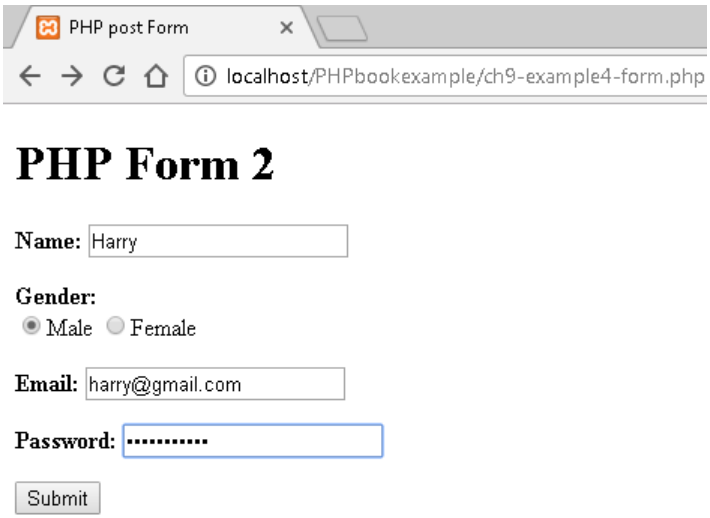

Figure 9-7 *The output of Example 4 with data in form*

In Figure 9-7 the data is filled. Now the user will submit this data and then the data will be sent to *ch9-example4-formdata.php* program. This program will receive the data and the output will be displayed as shown in Figure 9-8.

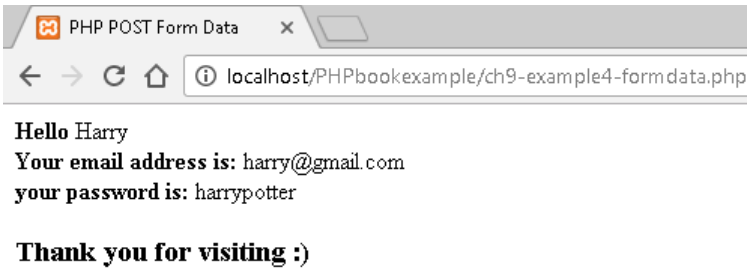

Figure 9-8 *The output of Example 4 displaying form data*

isset()

The **isset()** is the function used to check whether the variable is holding a value or not. If a variable is empty or have a **NULL** value then the **isset()** returns **false** otherwise **true**. The syntax of function **isset()** is as follow:

```
isset($var)
```

In the given syntax, **isset()** is the function and **$var** represents the variable. The **isset()** function will check whether the **$var** variable is holding a value or not.

Evaluation Copy. Do not reproduce. For information visit www.cadcim.com

trim()

The **trim()** function is used to remove the whitespace and other characters that are passed as parameter to the function, from the start and end point of a string. The syntax of function **trim()** is as follow:

```
trim($string,characters)
```

In the given syntax, **trim()** is the function containing two parameters where **$string** represents the variable holding string type value and **characters** represents any alphabet that is to be removed. The **$string** is the required parameter and **characters** is an optional parameter.

For example:

```
<?php
$str = "Hello User!";
echo trim($str,"Her!");
?>
//Output: llo Us
```

In the given example, **trim()** function will remove the **H**,**e**,**r**, and **!** characters from the **Hello User!** string assigned to **$str** variable and display the output **llo Us**.

stripslashes()

The **stripslashes()** function is used to remove the backslash(\) from the string. The syntax of function **stripslashes()** is as follow:

```
stripslashes($string)
```

In the given syntax, **stripslashes()** is the function containing a parameters **$string**. Here, **$string** represents the variable holding string type value.

For example:

```
<?php
$str = "Hello Ma\'am!!";
echo stripslashes($str);
?>
//Output: Hello Ma'am!!
```

In the given example, **stripslashes()** function will remove the backslash (\) from the **Hello Ma\'am!!** string assigned to **$str** variable and display the output **Hello Ma'am!!**.

Evaluation Copy. Do not reproduce. For information visit www.cadcim.com

htmlspecialchars()

The **htmlspecialchars()** function is used to convert the HTML special characters such as **<**, **>** , **&**, **'** , and **"** into HTML entities. This means that the HTML special characters such as HTML tags will be treated as normal characters. The syntax of function **htmlspecialchars()** is as follow:

```
htmlsecialchars($string)
```

In the given syntax, **htmlspecialchars()** is the function containing a parameter **$string**. Here, **$string** represents the variable holding string type value which may contain HTML special characters.

For example:

```
<?php
$txt = "<h1> and </h1> are the start and end tags of HTML heading
   element.";
echo htmlspecialchars($txt);
?>
//Output: <h1> and </h1> are the start and end tags of HTML heading
   element.
```

In the given example, **htmlspecialchars()** function will convert the **<h1>** and **</h1>** tags into normal characters. Next, it will display the **<h1>** and **</h1>** tags in the browser by using the **echo** statement instead of performing the HTML heading element operation.

THE $_SERVER Superglobla Variable

The **$_SERVER** superglobal variable is used in the PHP script to get the entries in the array from the web server. The syntax of using**$_SERVER** superglobal is as follow:

```
$_SERVER['ELEMENT']
```

In the given syntax, **$_SERVER[]** is an array variable. Here, **ELEMENT** represents the predefined server elements, for example, **PHP_SELF** , **REQUEST_METHOD**, **SERVER_NAME** and so on.

Example 5

In this example, you will create two programs, *ch9-example5.php* and*ch9-example5-formdata.php*, where the *ch9-example5.php* program will create a dynamic form and send the form data to *ch9-example5-formdata.php* program file using the **post** method. The *ch9-example5-formdata.php* program file will receive the form data from *ch9-example5.php* program file by using the **$_POST** and **$_SERVER** superglobal variables and will check that data by using the **trim()**, **stripslashes()**, and **htmlspecialchars()** functions to display the output in the browser.

The *ch9-example5.php* program file.

```
<!Doctype html>
<html>
<head>
```

Evaluation Copy. Do not reproduce. For information visit www.cadcim.com

```
<title>PHP Form 3</title>
<style>
   .err {
         color: red;
         }
   div {
         border: double 5px black;
         padding: 25px;
         }
   </style>
</head>
<body>
<div>
   <h2>PHP Form 3</h2>
   <p><span class="err">* represents required field.</span></p>
   <form method="post" action="ch9-example5-Formdata.php">
   Name: <input type="text" name="name" required>
   <span class="err">*</span>
   <br><br> Email:
   <input type="email" name="email" required>
   <span class="err">*</span>
   <br><br> Comment:
   <textarea name="comment" rows="5" cols="40"></textarea>
   <br><br> Gender:
<input type="radio" name="gender" value="Female" required>
Female
<input type="radio" name="gender" value="Male" required>
Male
<span class="err">*</span><br><br>
<input type="submit" name="submit" value="Submit">
         </form>
   </div>
</body>
</html>
```

The *ch9-example5-formdata.php* program file.

```
<!Doctype html>
<html>
<head>
<title>PHP Form 3 Data</title>
</head>
<body>
<?php
$name = $email = $gender = $comment = "";
  if ($_SERVER['REQUEST_METHOD'] == "POST") {
  $name = check_data($_POST["name"]);
  $email = check_data($_POST["email"]);
```

Evaluation Copy. Do not reproduce. For information visit www.cadcim.com

```
    $comment = check_data($_POST["comment"]);                    12
    $gender = check_data($_POST["gender"]);                      13
  }                                                              14
  function check_data($fdata) {                                  15
    $fdata = trim($fdata);                                       16
    $fdata = stripslashes($fdata);                               17
    $fdata = htmlspecialchars($fdata);                           18
    return $fdata;                                               19
  }                                                              20
  echo "<h2>Your Details:</h2>";                                 21
  echo $name;                                                    22
  echo "<br>";                                                   23
  echo $email;                                                   24
  echo "<br>";                                                   25
  echo $comment;                                                 26
  echo "<br>";                                                   27
  echo $gender;                                                  28
  ?>                                                             29
  </body>                                                        30
  </html>                                                        31
```

Explanation (ch9-example5.php)

Line 5

<style>

In this line, **<style>** is the start tag of HTML used to start the internal CSS of the web page.

Line 6 to Line 8

.err {

color: red;

}

In these lines, **.err** is the class name (selector) which is to be styled. Here, inside the block of selector starting with the { and ending with }, **color** is the property used to color the HTML text, and **red** is the value of the property.

Line 9 to Line 12

div {

border: double 5px black;

padding: 25px;

}

In these lines, **div** is the tag name (selector) which is to be styled. Here, inside the block of selector, **border** is the property used to give the border to an element, it has different border values where **double** is the border style, **5px** is border width, and **black** is border color. Next, **padding** is another property given to the **div** element to create **25px** space from all sides between the **border** and elements of **div**.

Evaluation Copy. Do not reproduce. For information visit www.cadcim.com

Line 13
</style>
In this line, **</style>** is the end tag of HTML used to end the internal CSS of the web page.

Line 18
** * represents required field. **
In this line, **** is the start tag of HTML used to add some properties to the HTML content. Here, **class** is an attribute used to name an element through which users can add style to that element, as done in Line 6 to Line 8 and **err** is a class name given to the **** element.

Line 27
<textarea name="comment" rows="5" cols="40"></textarea>
In this line, **name**, **rows**, and **cols** are three attributes given to the **<textarea>** element that has different values. Here, **rows** and **cols** defines the area of the textarea input box where **rows** is for number of rows and **cols** is for number of columns.

Explanation (ch9-example5-formdata.php)

Line 8
$name = $email = $gender = $comment = "";
In this line, **$name**, **$email**, **$gender**, and **$comment** variables are declared and no value is assigned to them.

Line 9
if ($_SERVER['REQUEST_METHOD'] == "POST") {
In this line, if condition is **$_SERVER['REQUEST_METHOD'] == "POST"**. Here **$_SERVER** superglobal variable has **REQUEST_METHOD** which is used to fetch the request method used to access the form data. Next, the condition is checked whether the request method is **POST** method or not. Here, the condition will get satisfied and the control will be transferred inside the **if** block and all statements (from Line 10 to Line 13) associated with the **if** block will be executed.

Line 10
$name = check_data($_POST["name"]);
In this line, **$name** is the variable which holds the **check_data()** function defined in Line 15 to Line 20. The **check_data()** function has **$_POST[]** superglobal variable which is used to fetch the data of **name** field from the form.

The working of Line 11 to Line 13 is similar to Line 10

Line 16
$fdata = trim($fdata);
In this line, **trim()** is the inbuilt function declared inside the user-defined function **check_data().** This line will receive the data in **$fdata** variable from the **If** control structure and perform the **trim()** functionality on the received data.

Line 17
$fdata = stripslashes($fdata);
In this line, **stripslashes()** is the inbuilt function declared inside the user-defined function **check_data().** This line will receive the data in **$fdata** variable from the **If** control structure and perform the **stripslashes()** functionality on the received data.

Evaluation Copy. Do not reproduce. For information visit www.cadcim.com

Line 18
$fdata = htmlspecialchars($fdata);
In this line, **htmlspecialchars()** is the inbuilt function declared inside the user-defined function **check_data().** This line will receive the data in **$fdata** variable from the **If** control structure and perform the **htmlspecialchars()** functionality on the received data.

Line 19
return $fdata;
In this line, **return** will return the data of **$fdata** after performing all the functionality on the received data.

The output of Example 5 is displayed in Figure 9-9.

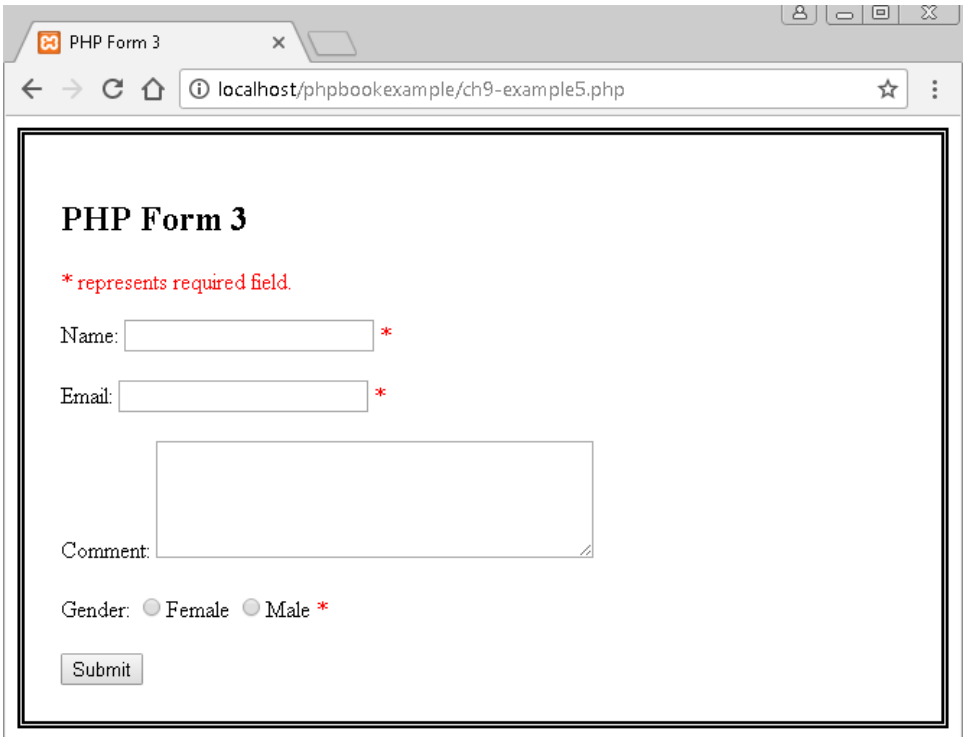

Figure 9-9 *The output of Example 5*

In Figure 9-9, the form is blank. Now, the user will enter the data to send it to another page as shown in Figure 9-10.

Evaluation Copy. Do not reproduce. For information visit www.cadcim.com

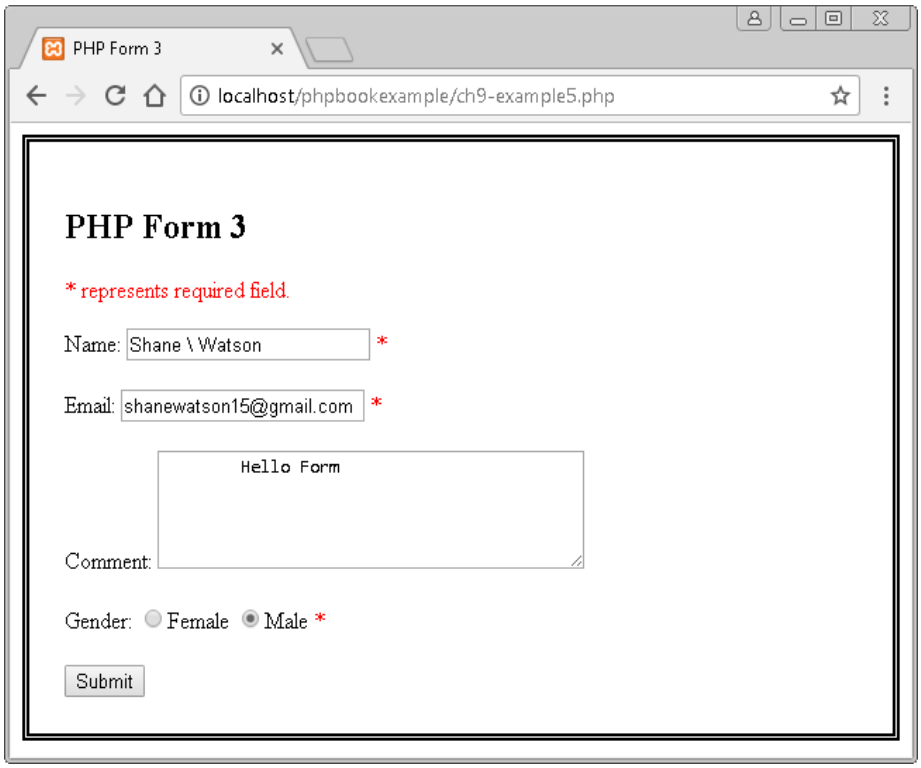

Figure 9-10 *The output of Example 5 with data in form*

In Figure 9-10 the data is filled. Now, the user will submit this data and the data will be sent to *ch9-example5-formdata.php* program. This program will receive the data and the output will be displayed as shown in Figure 9-11.

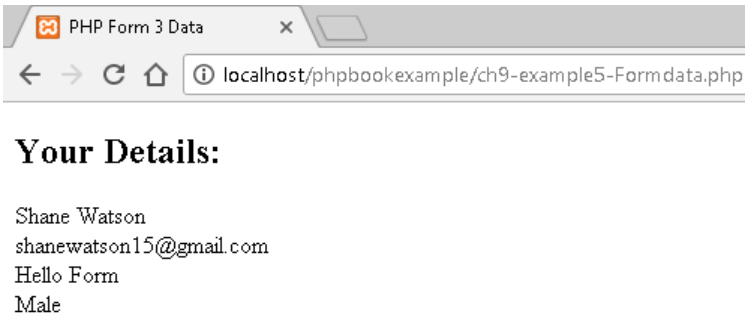

Figure 9-11 *The output of Example 5 displaying form data*

JavaScript(JS)

JavaScript is a lightweight object oriented scripting language which is used both for client-side scripting and server-side scripting. When JS is used to control the browser, to create dynamically updating content, validating form, and to control the Document Object Model, for example,

Evaluation Copy. Do not reproduce. For information visit www.cadcim.com

generating response to the user events when user provides any input to the form or clicks on button then the JS is working as client-side scripting language. When JS is used to communicate with a database or to manipulate files on server then the JS is working as server-side scripting language. JS is a case-sensitive language.

JS with HTML

The client-side scripting is done for the web browser by using JavaScript(JS) with HTML tags. The syntax of placing JS in HTML code is as follow:

```
<script>
         //JS code
</script>
```

In the given syntax, **<script>** is the start tag of HTML which is used to start the JavaScript code and **</script>** is the end tag of HTML which is used to end the Javascript code. The JavaScript code is placed between these tags.

You can place this syntax in the head section or in the body section of the HTML document. You can also create an external JS file and call it in the HTML document by following the syntax given next.

```
<script src="filename.js"></script>
```

In the given syntax, **src** is the attribute used to define the source path of the JS file inside the **<script>** tag and **filename.js** represents the complete path of JS file including the filename with its extension.

JS OUTPUT

You have already learned in the earlier chapters that to display the result or output of PHP script in the browser, PHP provides **echo** and **print** statements. Similarly, in JS there are four different way to display the output in the browser which are as follows:

1. **innerHTML**
2. **document.write()**
3. **window.alert()**
4. **console.log()**

innerHTML

The keyword **innerHTML** is used to access the HTML element to display the data declared inside the JS code. The syntax to use **innerHTML** is as follow:

```
document.getElementById('id').innerHTML = data;
```

In the given syntax, **document.getElementById('id')** is the method used to get the HTML element from the HTML document based on the element id. Here, **id** is the parameter passed inside the method which represents the value of the **id** attribute of an element that is to be accessed. Next, **innerHTML** is used with the method to access the element and display the **data** assigned to it where **data** represents a data of any type.

Evaluation Copy. Do not reproduce. For information visit www.cadcim.com

Example 6

The following program illustrates the use of **innerHTML** of JS. The program will do the sum of two numbers and display the output in the browser.

```
<!Doctype html>                                                    1
<html>                                                             2
<head>                                                             3
<title>JavaScript innerHTML</title>                                4
</head>                                                            5
<body>                                                             6
<h1>First JS program</h1>                                          7
   <p>The sum of two numbers- 11 and 21 is</p>                     8
   <p id="p1"></p>                                                 9
   <p><b id="p2"></b></p>                                          10
   <script>                                                        11
         document.getElementById("p1").innerHTML = 11 + 21;        12
         document.getElementById("p2").innerHTML = "The            13
         output is displayed";
   </script>                                                       14
</body>                                                            15
</html>                                                            16
```

Explanation

Line 9

** **

In this line, **<P>** and ** ** are the start and end tags of HTML paragraph that has attribute **id** whose value is **p1**.

Line 11

<script>

In this line,**<script>** is the start tag of HTML used to add the JavaScript in the program.

Line 12

document.getElementById("p1").innerHTML = 11 + 21;

In this line, **document.getElementById()** is the method where **p1** is the id of a paragraph declared in Line 9. Next, **innerHTML** is used to display the sum of 11 and 21 inside the HTML paragraph element in the browser.

Line 13

In this line, **document.getElementById()** is the method where **p2** is the id of a bold element declared in Line 10. Next, **innerHTML** is used to display the string inside the HTML element in the browser.

Line 14

</script>

In this line,**<script>** is the end tag of HTML used to end the JavaScript in the program.

Evaluation Copy. Do not reproduce. For information visit www.cadcim.com

The output of Example 6 is displayed in Figure 9-12.

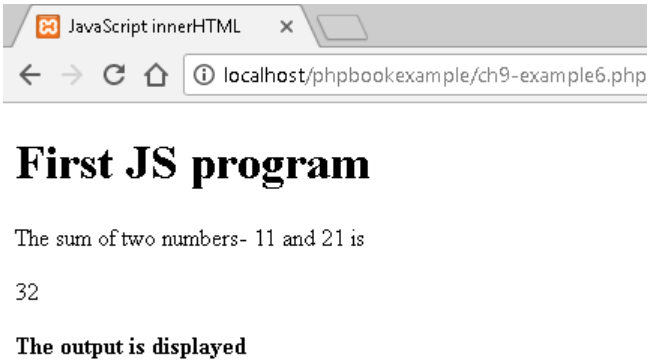

Figure 9-12 *The output of Example 6*

document.write()

The **documemt.write()** method of JS is used to write the content or HTML expressions for the document and display it on the web. It is mostly used to test the working of web page. The syntax to use **documemt.write()** of JS is as follow:

```
document.write(expression1, expexpression2,...)
```

In the given syntax, **documemt.write()** is the method used to write expressions in the document using JS where **expression1**, **expression2** represent multiple expressions that can be passed inside the method. These expressions will be displayed in the browser.

For example:

```
<script>
   document.write("Sum is <br>");
   document.write(11 + 21);
</script>
/*output: Sum is
          32 */
```

In the given example, **document.write()** will display the string **Sum** is and break the line in the browser because of the **
** tag. Next, **document.write()** will perform the addition operation and display the resultant value 32 in the browser.

window.alert()

The **window.alert()** method of JS is used to write the alert message for the document and display it in the alert box on the web which has the **ok** button. To proceed further, you will have to click on **ok** button of the alert box. The syntax to use **window.alert()** of JS is as follow:

```
window.alert("msg")
```

In the given syntax, **window.alert()** is the method used to write the alert message using JS where **msg** represents any message that is to be displayed in the alert box in the browser.

Evaluation Copy. Do not reproduce. For information visit www.cadcim.com

Example 7

The following program illustrates the use of **window.alert()** method of JS. The program will display the message in the alert box as output in the browser.

```
<!Doctype html>                                                    1
<html>                                                             2
<head>                                                             3
<title>window alert box</title>                                    4
</head>                                                            5
<body>                                                             6
   <h2> You saw the alert box.</h2>                                7
   <script>                                                        8
         window.alert("This is alert box msg");                    9
   </script>                                                       10
</body>                                                            11
</html>                                                            12
```

Explanation

Line 9

window.alert("This is alert box msg");

In this line, **window.alert()** method is used to display the message passed inside its parentheses in the alert box as output.

The output of Example 7 showing alert box is displayed in Figure 9-13.

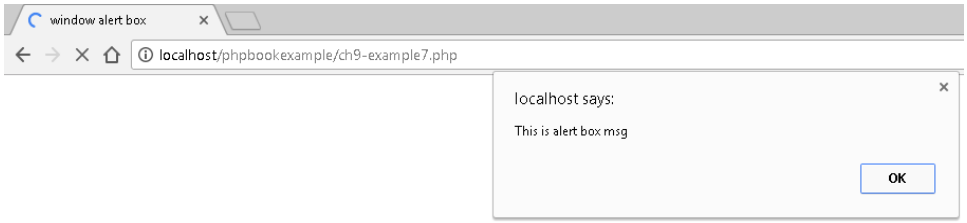

Figure 9-13 *The output of Example 7 displaying alert box*

Next, when the user will click the **ok** button of the alert box the other content of the web page defined in the program will be displayed as shown in figure 9-14.

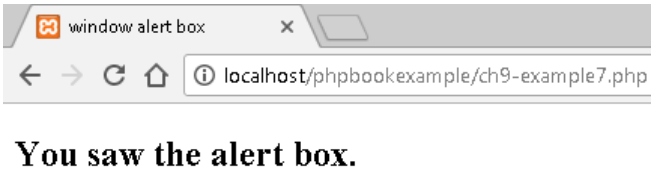

Figure 9-14 *The output of Example 7 displaying the content*

Evaluation Copy. Do not reproduce. For information visit www.cadcim.com

console.log()

The **console.log()** method of JS is used to write and display any expression passed to it in the **Console** tab of the browser. It is used for debugging. Every browser has different path to reach the **Console** tab. The **Console** tab in the browser displays all the errors and other messages. The syntax to use **console.log()** of JS is as follow:

```
console.log(expression)
```

In the given syntax, **console.log()** is the method used to debug the code where **expression** represents any message or expression passed to it that is to be displayed in the **Console** tab in the browser.

The **Console** tab is not visible directly in the browser. It is displayed in different display box in the browser. For example, to open the **Console** tab in the Google Chrome browser, you can press **F12** button on the keyboard or you can right-click on the browser and then select **Inspect** option where you will find the **Console** tab in the separate display box in the browser.

Example 8

The following program illustrates the use of the **console.log()** method of JS. The program will display the message in the **Console** tab as output in the browser.

```
<!Doctype html>                                                  1
<html>                                                           2
<head>                                                           3
<title>Console output</title>                                    4
</head>                                                          5
<body>                                                           6
<p>To open console tab in Google chrome press <b>F12</b></p>     7
   <script>                                                      8
         console.log("No error found");                          9
   </script>                                                     10
</body>                                                          11
</html>                                                          12
```

Explanation

Line 9

console.log("No error found");

In this line, the **console.log()** method is used to display the message or expression passed inside its parentheses in the **Console** tab in the browser.

The output of Example 8 is displayed in Figure 9-15.

Evaluation Copy. Do not reproduce. For information visit www.cadcim.com

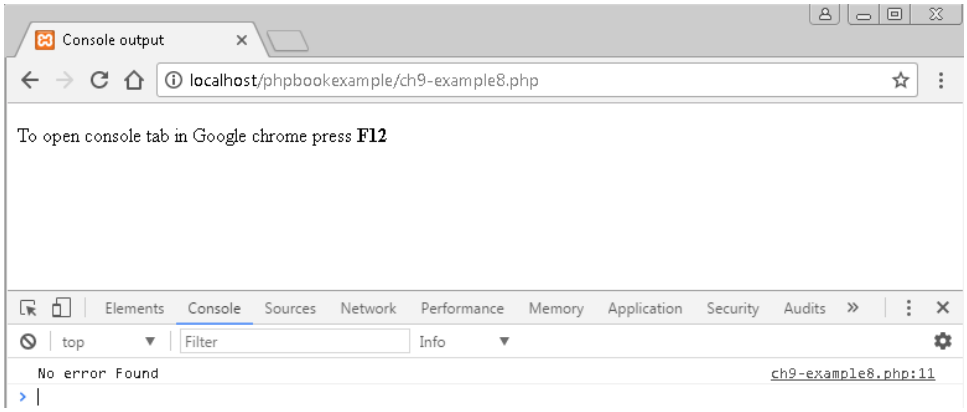

Figure 9-15 *The output of Example 8*

JavaScript Variables

The variables in JavaScript are declared by using the **var** keyword along with variable name. You can declare multiple variables at a time by separating them using comma.

For example:

```
<script> var a, b, c; </script>
```

In the given example, **var** is keyword used to declare the variable and **a**,**b**, and **c** are the variable names.

While naming a variable in JavaScript, you must follow certain rules. The rules are as follows:

1. Only alphabetic characters- both uppercase and lowercase, digits from 0 to 9, **$** symbol and underscore (_) can be used.
2. Variable name can start with an alphabet, **$**, or an underscore but not with a digit.
3. Variable name can not contain space. If variable name contain more than one word, it can be separated by using _ (underscore) character or can be written in camel case instead of using space.
4. Variable names are case sensitive. For example, **javascript** and **JavaScript** are two different variables, one in lowercase case and other in camel case.

Note

*If a value is assigned to the variable then it is optional to declare the variable with **var**. For example, **a = 5;** where **a** is a variable. But, to avoid errors declare all variables with **var**.*

Example 9

The following program illustrates the use of JavaScript variables. The program will use multiple JS variables to display the output in the browser.

```
<!Doctype html>                                                    1
<html>                                                             2
```

Evaluation Copy. Do not reproduce. For information visit www.cadcim.com

```
<head>                                                                  3
<title>JS variables</title>                                             4
</head>                                                                 5
<body>                                                                  6
<h3>You can declare many variables in one statement.</h3>               7
<p><i>Hobby</i></p>                                                     8
<p id="par"></p>                                                        9
<h3>To open console tab in Google chrome press F12</h3>                 10
   <script>                                                             11
         var v1;                                                        12
         var v2;                                                        13
         console.log(v1 = 5 * 10);                                      14
         v2 = v1 + 2;                                                   15
         document.write("<i>The sum is : </i>" + v2);                   16
         var name = "James Bond", age = 53, hobby = "Acting";           17
         document.getElementById("par").innerHTML = hobby;              18
   </script>                                                            19
</body>                                                                 20
</html>                                                                 21
```

Explanation

Line 13 and Line 14

var v1;

var v2;

In these lines, **v1** and **v2** are the JS variable names declared by using keyword **var.**

Line 14

console.log(v1 = 5 * 10);

In this line, **v1** variable is holding an expression where multiplication of 5 and 10 is done with the help of the ***** operator. The output of this line will be displayed in **Console** tab in the browser.

Line 16

document.write("The sum is : " + v2);

In this line, concatenation between the string and the variable passed inside the **document.write()** method is done with the **+** operator. This line will display the following in the browser.

The sum is : 52

Line 17

var name = "James Bond", age = 53, hobby = "Acting";

In this line, multiple variables are declared in one line by using the **var** keyword and each variable is holding a value.

The output of Example 9 is displayed in Figure 9-16.

Evaluation Copy. Do not reproduce. For information visit www.cadcim.com

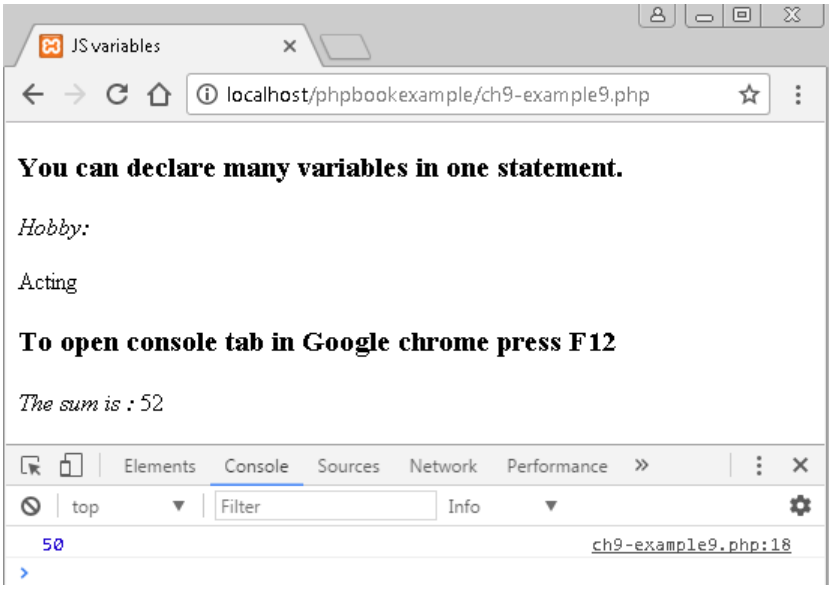

Figure 9-16 *The output of Example 9*

FORM VALIDATION USING JS

JavaScript is used for form validation to validate the user input to get the correct data instead of junk values. To validate any form, it is important to match every character that the user enters in a form field with the set of regular expression supported by both JS and PHP.

Regular Expression

Regular expressions are used to match the specified pattern or characters with the user input data. With the help of regular expressions, you can validate any form. It is helpful in construction of a pattern matching algorithms in a single expression. Every regular expression are enclosed in pair of forward slashes.

For example:

```
/Hello *User/
```

In the given example, **/Hello *User/** is a regular expression where **Hello *User** is a pattern to be matched which is enclosed in pair of forward slashes.

Note
Regular expressions are case-sensitive.

There are different groups of character matching used in regular expression which are as follow:

1. Metacharacters
2. Character classes

Evaluation Copy. Do not reproduce. For information visit www.cadcim.com

Metacharacters

Metacharacters are characters which have special meanings when used as regular expression. Table 9-1 shows a list of some of the metacharacters with their description and example.

Table 9-1 *Metacharacters with their description*

Metacharacter	Description	Example
*	Asterisk (*) matches any number of preceding characters including no character.	To match **HelloUser** or **Hello User**, the regular expression **/Hello *User/** can be used.
+	It matches any number of preceding characters except empty case or no character.	To match **Hello User**, the regular expression **/hello +User/** can be used but you cannot match **HelloUser** with the same expression.
.	It matches any single character. If written with ***** or **+** metacharacter it will match every character.	To match **bye**,**bee**, or **b2e**, the regular expression **/<b.e>/** can be used.
\.	It matches the dot(**.**) symbol instead of reading it as metacharacter.	To match dot (**.**) symbol in string **gmail.com**, the regular expression **/\./** is used.
\+	It matches the + symbol instead of reading it as metacharacter.	To match **+** symbol in the string **5 + 7**, the regular expression **/\+/** is used.
*	It matches the asterisk(*****) symbol instead of reading it as metacharacter.	To match ***** symbol in the string **5 * 7**, the regular expression **/*/** is used.
\s	It matches a whitespace character.	To match the whitespace in the string **How are you**?, the regular expression **/\s/g** will be used where **g** is a modifier.
\S	It matches a nonwhitespace character.	To match every character other then space in the string **How are you?**, the regular expression **/\S/g** will be used where **g** is a modifier.
\w	It matches a word character (a-z,A-Z,0-9, and _).	To match the word character in any string **/\w/** is used.
\W	It matches anything other than word character (a-z,A-Z,0-9, and _).	To match the word character in any string **/\W/** is used.
\d	It matches a single digit.	To match only numbers such as 1, 2, 3 and so on, the regular expression **/\d/** is used.
\D	It matches a single non-digit.	To match characters other then numbers, the regular expression **/\D/** is used.
{n}	It matches anything **n** number of times.	To match 4 digit numbers such as 0000, 2000, and so on, the regular expression **/\d{4}/** is used.

Evaluation Copy. Do not reproduce. For information visit www.cadcim.com

Character Classes

Character classes are used in regular expression to match one or more characters defined inside the character class. If any of the character defined inside the character class appears, the data matches. The character class starts with **[** and ends with **]** in a regular expression. You can place a list of characters inside the pair of square brackets **[ ]**. But, it matches exactly one character out of the list of characters. For example, if you want to match both **sun** and **son** then you can specify the regular expression as follow:

```
/s[ou]n/
```

In the given example, the regular expression can match **son** as well **sun** but not **soun** or **suon**. Because it matches exactly only one character out of the character list of character class. In this case, it can match either **o** or **u** but not **ou** or **uo** from the character class.

Character classes can also be used to define a range of characters from which a single data can appear in the data to be matched. To define a range, you can use the hyphen (**-**) inside the character class between two characters. For example, to match a single digit, you can use the regular expression **/[0-9]/**. This expression can match a single digit from the range 0 to 9.

If you do not want to match the characters given in the character class then you will have to use ^ symbol before that character inside the character class. For example, **/[^a-z]/** regular expression will match every character other than **a** to **z** characters.

Regular Expression Modifiers

Regular expression modifiers are used with regular expressions to modify them. These modifiers provides additional functionality to the regular expression. There are 3 regular expression modifiers listed in Table 9-2 with their description.

Table 9-2 *Modifiers with their description*

Modifiers	Description	Example
g	Globally search and matches the characters. This will search all matches.	**/girls/g** will match all the occurrence of word **girls** in the string.
i	Enables case-insensitive matching.	**/girls/i** will match **girls** as well as **Girls** in the string.
m	Search and matches the characters in multiple lines. If ^ (caret) is placed with the string inside the regular expression that has **m** modifier then it will match only at start of the each line. Similarly, if **$** is placed then it will match at the end of the each line.	**/^Hello/m** will match Hello in string **\nHello user\n Hello friends**. Here, **\n** is used for next line.

Evaluation Copy. Do not reproduce. For information visit www.cadcim.com

Example 10

In this example, you will create two programs, *ch9-example10.php* and *display.php*, where the *ch9-example10.php* program will create a dynamic form and validate the form by using JavaScript and send the form data to *display.php* program file using the **post** method. The *display.php* program file will receive the form data from *ch9-example10.php* program file to display the output in the browser.

The *ch9-example10.php* program file.

```
<!Doctype html>
<html>
<head>
<title>JS validate</title>
<style>
        .err {color: red;}
        div {
              border: double 5px black;
              padding: 25px;
              margin-right: 1000px;
            }
   </style>
   <script>
        function validateform(form) {
          invalid = validateFname(form.firstname.value);
          invalid += validateLname(form.lastname.value);
          invalid += validateUsername(form.username.value);
          invalid += validatePassword(form.password.value);
          invalid += validateAge(form.age.value);
          invalid += validateEmail(form.email.value);
              if (invalid == "") return true;
              else {
                    alert(invalid);
                    return false;}}
   function validateFname(field){
        return (field == "") ? "First Name Required.\n" : "";}
   function validateLname(field){
        return (field == "") ? "Last name Required.\n" : "";}
   function validateUsername(field){
        if (field == "") return "Username Required.\n";
              else if (field.length < 5)
        return "Username must be at least 5 characters.\n";
              else if (/[^a-zA-Z0-9_-]/.test(field))
   return "Only a-z,A-Z,0-9,-,and _ allowed in Username.\n";
   return "";}
   function validatePassword(field){
        if (field == "") return "Password Required.\n";
              else if (field.length < 6)
```

Evaluation Copy. Do not reproduce. For information visit www.cadcim.com

```
            return "Password must be at least 6 characters.\n";
                  else if (!/[a-z]/.test(field) ||
                        !/[A-Z]/.test(field) ||
                        !/[0-9]/.test(field))
      return "Password require one each of a-z, A-Z, and 0-9.\n";
      return "";}
      function validateAge(field){
            if (isNaN(field)) return "Age Required.\n";
                  else if (field < 18 || field > 110)
            return "Age must be between 18 and 110.\n";
            return "";}
      function validateEmail(field){
            if (field == "") return "Email Required.\n";
                  else if (!((field.indexOf(".") > 0) &&
                              (field.indexOf("@") > 0)) ||
                        /[^a-zA-Z0-9.@_-]/.test(field))
            return "The Email address is invalid.\n";
            return "";}
      </script>
   </head>
   <body>
   <div>
      <h2>JS validate PHP Form</h2>
      <p><span class="err">* represents required field.</span></p>
      <form method="post" action="display.php"
            onsubmit="return validateform(this)">
   <p>First Name<input type="text" maxlength="32" name="firstname">
            <span class="err">*</span></p>
   <p>Last Name<input type="text" maxlength="32" name="lastname">
            <span class="err">*</span></p>
   <p>Username<input type="text" maxlength="16" name="username">
            <span class="err">*</span></p>
      <p>Password<input type="password" maxlength="12"
                  name="password">
            <span class="err">*</span></p>
      <p>Email<input type="email" maxlength="64" name="email">
            <span class="err">*</span></p>
      <p>Age<input type="text" maxlength="3" name="age">
            <span class="err">*</span></p>
      <input type="submit" value="Signup">
      </form>
      </div>
      </body>
      </html>
```

The *display.php* program file.

```
   <!Doctype html>
   <html>
```

Evaluation Copy. Do not reproduce. For information visit www.cadcim.com

```
<head>                                                                  3
<title>JS validate Data</title>                                         4
</head>                                                                 5
<body>                                                                  6
<?php                                                                   7
$fname = $lname = $uname = $email = $password = $age                    8
= $gender = "";
   if ($_SERVER['REQUEST_METHOD'] == "POST") {                          9
  $fname = check_data($_POST["firstname"]);                             10
  $lname = check_data($_POST["lastname"]);                              11
  $uname = check_data($_POST["username"]);                              12
  $email = check_data($_POST["email"]);                                 13
  $password = check_data($_POST["password"]);                           14
   $age = check_data($_POST["age"]);         }                          15
function check_data($fdata) {                                           16
  $fdata = trim($fdata);                                                17
  $fdata = stripslashes($fdata);                                        18
  $fdata = htmlspecialchars($fdata);                                    19
  return $fdata;                                                        20
                }                                                       21
echo "<h2>Your Details:</h2>";                                          22
echo "Your first name is <b>$fname</b>";                                23
echo "<br>";                                                            24
echo "Your last name is <b>$lname</b>";                                 25
echo "<br>";                                                            26
echo "Your username is <b>$uname</b>";                                  27
echo "<br>";                                                            28
echo "Your email id is <b>$email</b>";                                  29
echo "<br>";                                                            30
echo "Your password is <b>$password</b>";                               31
echo "<br>";                                                            32
echo "Your age is <b>$age</b>";                                         33
?>                                                                      34
</body>                                                                 35
</html>                                                                 36
```

Explanation (ch9-example10.php)

Line 13

<script>

In this line, **<script>** is the start tag used to start the JavaScript code.

Line 15

invalid = validateFname(form.firstname.value);

In this line, the **validateFname() function** is called to validate the form input field where **invalid** is a variable name, the **validateFname()** function is an user-defined function, and **form.firstname.value** is the parameter passed to the function. This parameter follows the Document Object Model (DOM) which breaks down the parts of an HTML into discrete objects. Here, **form** is used to call the HTML form tag **<form>**, **firstname** is the value of **name** attribute in the form input field, and **value** is for the user input value.

Evaluation Copy. Do not reproduce. For information visit www.cadcim.com

The working of Line 16 to Line 20 is similar to Line 15.

Line 21 to Line 24

```
if (invalid == " ") return true;
else {
        alert(invalid);
        return false;}}
```

In these lines, the **if** statement will check the condition that the **invalid** variable is empty or not. If it is empty, then it will return **true** otherwise an alert box will pop up that displays the string which contains the error message as specified in the functions.

Line 25 to Line 26

```
function validateFname(field) {
        return (field == "") ? " First Name Required.\n" : "";}
```

In these lines, the **validateFname()** function is testing whether the value in the **field** parameter is blank or not. The **field** parameter is receiving the value from **firstname** passed to the **validateFname()** function (Line 15). If the value of the **field** parameter is empty then the string **First Name Required.** is returned to the **validateform()** function and the string get displayed in the alert box. Otherwise, it returns an empty string to the **validateform()** function which signifies that no error is found in the received value.

The working of Line 27 to Line 28 is similar to the Lines 25 to 26.

Line 29 to Line 30

```
function validateUsername(field) {
   if (field == "") return "Username Required.\n";
```

In these lines, the **validateUsername()** function is testing whether the value in the **field** parameter is blank or not. The **field** parameter is receiving the value from **username** passed to the **validateUsername()** function (Line 17). If the value of the **field** parameter is empty then the string **Username Required.** is returned to the **validateform()** function and the string gets displayed in the alert box. Otherwise, the control will be transfered to Line 31.

Line 31 to Line 32

```
else if (field.length < 5)
return "Username must be at least 5 characters.\n";
```

In these lines, the **else if** condition is evaluated. If the length of the value received from the **validateUsername()** function (Line 17) is less than 5, then the next line (Line 32) will return the string **Username must be at least 5 characters.** to the **validateform()** function. Otherwise, the control will be transfered to Line 33.

Line 33 to Line35

```
else if (/[^a-zA-Z0-9_-]/.test(field))
        return "Only a-z, A-Z, 0-9, -, and _ allowed in Username.\n";
return "";
```

Evaluation Copy. Do not reproduce. For information visit www.cadcim.com

In these lines, the **else if** condition is evaluated where **/[^a-zA-Z0-9_-]/** is the regular expression and **test()** method is the JS inbuilt method which will test the **field**. The **else if** condition will check the value received from the **validateUsername()** function (Line 17). Here, if any character other than **a** to **z**, **A** to **Z**, **0** to **9**, **_**, and **-** exist then the control will be transfered to the next line (Line 34) where the string **Only a-z, A-Z, 0-9, - and _ allowed in Username.** is returned to the **validateform()** function. Otherwise, the control will be transfered to Line 35 which returns the empty string to the **validateform()** function which signifies that no error is found in the received value.

Line 40 to Line 41

```
else if (!/[a-z]/.test(field) ||!/[A-Z]/.test(field) ||!/[0-9]/.test(field))
return "Passwords require one each of a-z, A-Z, and 0-9.\n";
```

In these lines, the **else if** condition is evaluated. If the value received from the **validatePassword()** function (Line 18) does not contain at least one character each from **a** to **z**, **A** to **Z**, and **0** to **9** then the control will be transfered to the next line (Line 41) and the string P**assword require one each of a-z, A-Z, and 0-9.** is returned to the **validateform()** function. Otherwise, the control will be transfered to Line 42.

Line 44

```
if (isNaN(field)) return "Age Required.\n";
```

In this line, **isNaN()** is the JS inbuilt method used to check whether a value is a number or not. If the value of the **field** parameter is other than a number or is empty then the string **Age Required.** is returned to the **validateform()** function and the string get displayed in the alert box. Otherwise, the control will be transfered to Line 45.

Line 45 to Line 46

```
else if (field < 18 || field > 110)
        return "Age must be between 18 and 110.\n";
```

In these lines, the **else if** condition is evaluated. If the value received from the **validateAge()** function (Line 19) is less than 18 or greater than 110 then the control is transfered to the next line (Line 46) and the string **Age must be between 18 and 110.** is returned to **validateform()** function. Otherwise, the control will be transfered to Line 47.

Line 50 to Line 51

```
else if (!((field.indexOf(".") > 0) &&(field.indexOf("@") > 0))
        ||/[^a-zA-Z0-9.@]/.test(field))
return "The Email address is invalid.\n";
```

In these lines, the condition will check the value received from the **validateEmail()** function (Line 20). If the index position of **.** and **@** are not greater than **0** or the value contain any character other than **a** to **z**, **A** to **Z**, **0** to **9**, **.**, and **@** then the condition will be true and the next statement (Line 51) is executed where the string **The Email address is invalid.** is returned to **validateform()** function. Otherwise, the control will be transfered to Line 52.

Evaluation Copy. Do not reproduce. For information visit www.cadcim.com

Line 59
<form method="post" action="display.php" onsubmit="return validateform(this)">
In this line, **onsubmit** is an event attribute which executes the JavaScript function when the form is submitted by the user. Here, the **onsubmit** attribute will execute the **validateform()** function of Javascript.

Explanation (display.php)

The working of *display.php* program is similar to the *ch9-example5-formdata.php* program.

The output of Example 10 is displayed in Figure 9-17.

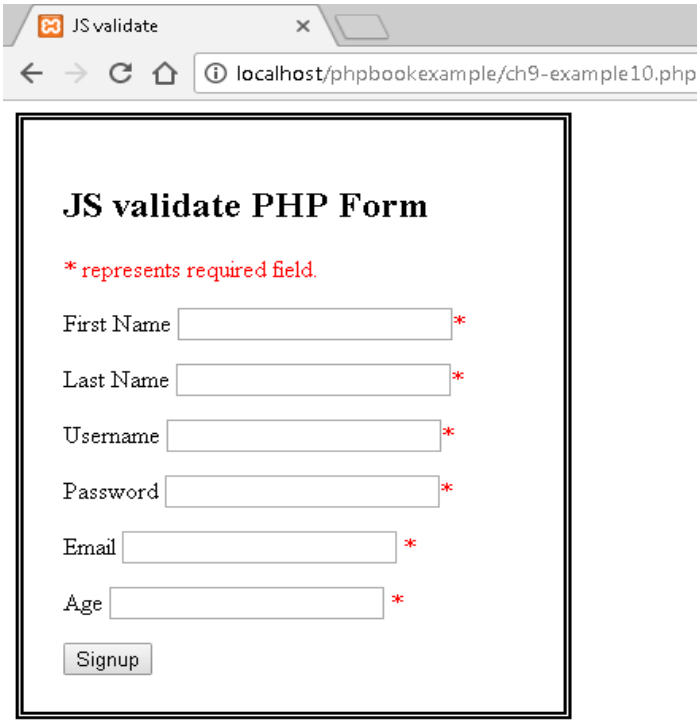

Figure 9-17 *The output of Example 10*

In Figure 9-17, the form is blank. Next, the user will enter the data and submit the data by clicking on **Signup** button in the form to send data to another page as shown in Figure 9-18.

Evaluation Copy. Do not reproduce. For information visit www.cadcim.com

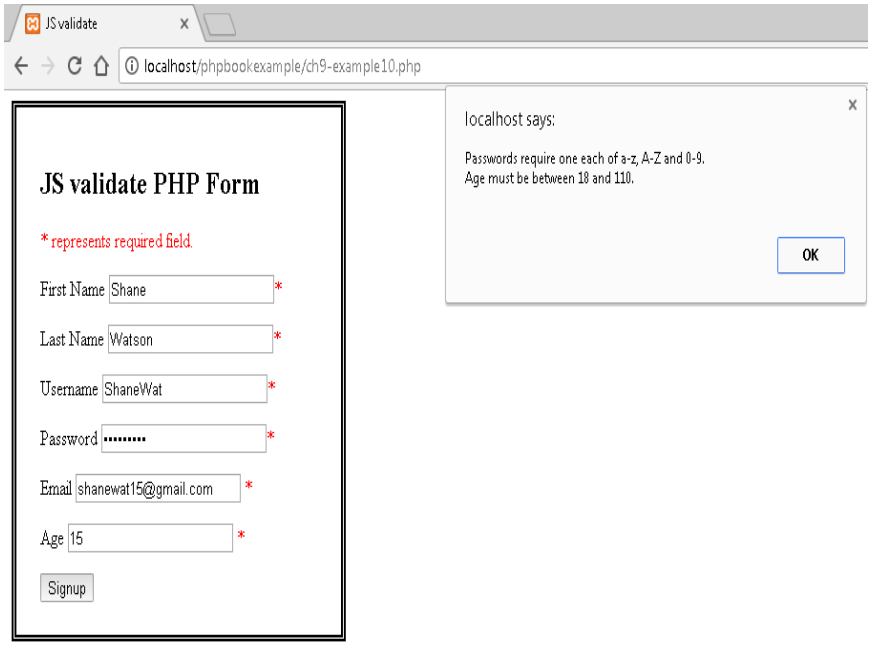

Figure 9-18 *The output of Example 10 displaying data in form and alert box*

In figure 9-18, the alert box is displayed because of JavaScript validation. The alert box in the Figure 9-18 displays the message to inform the user that the data entered is invalid. Now, the user will fill the data according to the validation rules set as shown in Figure 9-19.

Evaluation Copy. Do not reproduce. For information visit www.cadcim.com

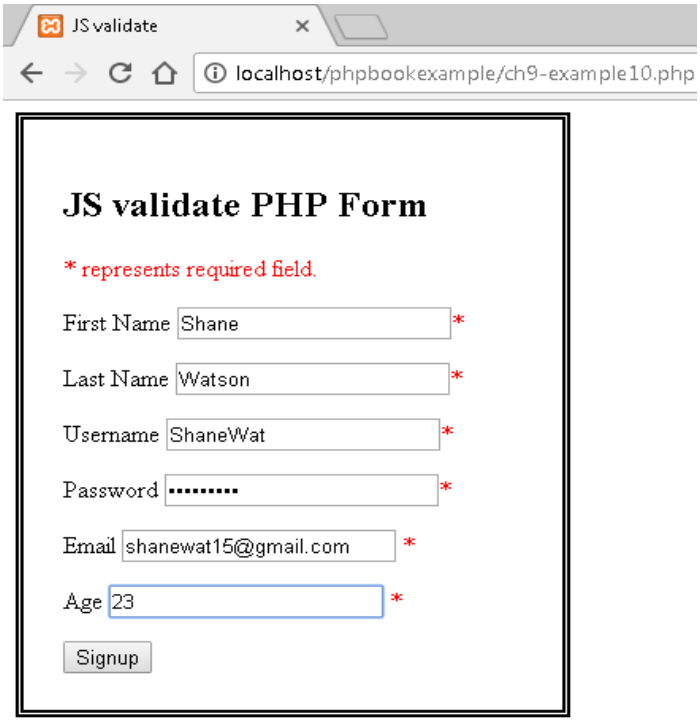

Figure 9-19 *The output of Example 10 displaying the valid form data*

In Figure 9-19 the data is filled. Now the user will again submit the form data by clicking on the **Signup** button. If the data is valid, the data will be sent to *display.php* program. This program will receive the data and the output will be displayed, as shown in Figure 9-20.

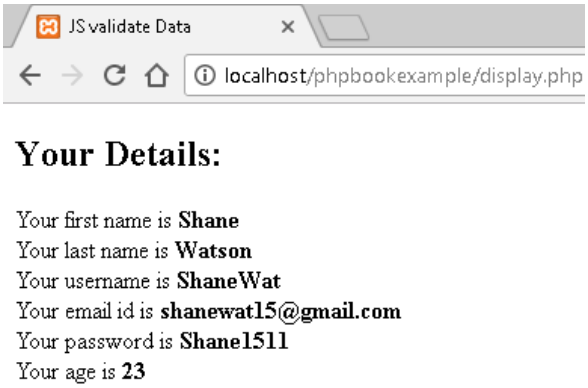

Figure 9-20 *The output of Example 10 displaying form data*

Regular Expression Functions in PHP

To use regular expression in PHP, you need to pass them as parameter to the regular expression function. The two most common regular expression functions are as follows:

Evaluation Copy. Do not reproduce. For information visit www.cadcim.com

1. preg_match()
2. preg_replace()

preg_match()

The **preg_match()** function is used to match a string with the help of regular expression. The syntax of **preg_match()** is as follows:

```
preg_match("/regularExp/","string",$match);
```

In the give syntax, **preg_match()** is the regular expression function and three parameters are passed to the **preg_match()** function. Here, **/regularExp/** represents regular expression which is to be matched with the text, **string** represents the text in which match is performed, and the third parameter passed to the function is **$match** which is optional, this represents the text that is matched.

For example:

```
<?php
preg_match("/php/i","Latest version of PHP is PHP 7",$result);
echo "$result[0]";
?>
//output: PHP
```

In the given example, the **preg_match()** function will match the first occurrence of the word **PHP** in the string **Latest version of PHP is PHP 7** and display **PHP** as output.

preg_replace()

The **preg_replace()** function is used to match and replace string with the help of regular expression. The syntax of **preg_replace()** is as follows:

```
preg_replace("/regularExp/","replacement","string");
```

In the given syntax, **preg_replace()** is the regular expression function and three parameters are passed to the **preg_replace()** function. Here, **/regularExp/** represents regular expression which is to be searched and replaced, **replacement** represents the replacement text, and **string** represents the string in which the replacement of text is done.

For example:

```
<?php
echo preg_replace("/php/i","HTML","PHP is a scripting language");
?>
//output: HTML is a scripting language
```

In the given example, the **preg_replace()** function will replace **PHP** with **HTML** in the string **PHP is a scripting language** and display **HTML is a scripting language** as output.

Evaluation Copy. Do not reproduce. For information visit www.cadcim.com

Self-Evaluation Test

Answer the following questions and then compare them to those given at the end of this chapter:

1. An _____ in HTML provides additional information to the HTML elements.
2. The form _____ elements are required to create the form input controls.
3. The _____ value provides one-line input text box.
4. The form datalist element in HTML is used to create _____ list.
5. The _____ method is the secure method to send the input data to another page.
6. The **isset()** function is used to check whether the variable is holding a value or not. (T/F)
7. The keyword **innerHTML** is used to access the HTML element to display the data declared inside the PHP code. (T/F)
8. The **g** modifier enables case-insensitive matching. (T/F)

Review Questions

Answer the following questions:

1. Which of the following is the correct syntax of form input element?

   (a) `<input field="value">` (b) `<input element="value">`
   (c) `<input attr="value">` (d) `<input type="value">`

2. Which of the following form **method** is used to transfer data using HTTP header?

   (a) **get** (b) **post**
   (c) **header** (d) **URL**

3. Which of the following form attributes is used to specify the page where the form data will be sent by using **get** or **post** method.?

   (a) **action** (b) **enctype**
   (c) **title** (d) **method**

4. Which of the following functions is used to check whether the variable is holding a value or not?

   (a) **trim()** (b) **isset()**
   (c) **htmlspecialchars()** (d) **stripslashes()**

Evaluation Copy. Do not reproduce. For information visit www.cadcim.com

5. Which of the following regular expression modifiers is used for case-insensitive matching?

(a) **g** (b) **i**
(c) **c** (d) **m**

6. Which of the following JS methods is used to write and display any expression passed to it in the **Console** tab of the browser.

(a) **window.alert()** (b) **Console.error()**
(c) **console.log()** (d) **document.write()**

7. Which of the following functions is used to match and replace string with the help of regular expression?

(a) **preg_replace()** (b) **match_replace()**
(c) **isset()** (d) **preg_match()**

EXERCISES

Exercise 1

Create a form containing 5 types of input fields and a datalist drop-down.

Exercise 2

Create a form having 5 input elements and validate the form using JavaScript to display the form data in the browser.

Evaluation Copy. Do not reproduce. For information visit www.cadcim.com

Answers to Self-Evaluation Test

1. attribute, **2.** input, **3. text**, **4.** drop-down, **5. post**, **6.** T, **7.** F, **8.** F