# SE-301L Web Engineering Lab

# Lab Manual


Prepared By:

**Engr. Muhammad Humayun Khan**

## Table of Contents

## Experiment No. 1 HTML Text Formatting Tags, Tables, Forms Controls, HTML 5 Elements

**Objectives:** To familiarize students with HTML basic tags, tables, forms, attributes and controls used for data input in web pages and HTML 5 different elements.

**Tools:** Notepad, Browser (Internet Explorer, Firefox or Google Chrome)

**Procedure:** Creating html prototype for signup page using the following html and html 5 elements:

- Table
- Forms
- Text Boxes
- Text Areas
- Radio Buttons
- Check Boxes
- Submit and Reset Buttons
- Color
- Audio
- Input Type: number
- Canvas

**The Hypertext Markup Language (HTML)** is the language of the World Wide Web. Every document in the Web is written in HTML, and all the document formatting, clickable hyperlinks, graphical images, jumping java applets, multimedia documents, fill-in-forms, and other many web activities and events you have seen are based on HTML and is not case sensitive. HTML is simple, easy to learn markup language.

**Tables** play a vital role in html document and helps in organizing the data in a decent way. The portions of a table or you can say as grid is called as a "Cell". The Table begins with the tag < Table > and ends with the tag < /Table >. Each row in the table is identified by the "Table row" tag < TR > and each column is identified by the "Table Data" tag < TD > and all has the closing tag. The attributes of the table are:

**< table border = n bgcolor = "any color" width = n% height = n% background = "url" cellpadding = n cellspacing = n align = left / right / center bordercolor = " any color " >**

- **Border:** shows the width of the table lines.
- **Width:** specify width of the table.
- **Height:** specify height of the table.
- **Background:** is used to insert image to the background of the table.
- **Cellpadding:** is used to give space between the right side of the cell and the text.
- **Cellspacing:** is used to give space between the cells of the table.

Sometimes inside a table we want to give a heading so then we use “table header” tag which is as <th>. It has two very important attributes that is "rowspan" and "colspan". These attributes are used for the purpose of merging rows and columns respectively.

**Forms** are used in web pages to allow communication between your viewers and your website, to gather information and to offer different means of navigation. The role of forms is to gather different information and wraps it up into a packaged format that can be sent directly to a web server where there is a customized program sitting and waiting for the form information. These program can unpackaged the information, manipulate it, store data and send feedback page back to the viewer.

A web page form is defined by a set of **< form >.....< /form >** tags where everything is included as text fields, buttons, checkboxes etc. The format for writing a form is:

**< form method = "get / post" > form elements < /form >**

The two values for method attributes are, "get and post". Both are used to send data to the server but the difference is that "get" is used when we want to send limited data to a server that is of 1024kb and "post" is used when we want to send more data to a server. Different form elements are explained below:

The **Menu Select** option provides drop-down menus that allow the viewer to choose one from a list of choices. The **< option >...< /option >** tag defines the text that is displayed in the menu. The html format is:

```
< select name = "any name" >
 < option value = "any value" > any text < /option >
                .....
                .....
                 .....
< /select >
```

Multiple elements from the menu can be selected by writing the keyword "multiple" in the select tag along with its size that how many you want to select.

The **text input** is a one line field where user can enter some data as required. The html and attributes of input text are:

**< input type = "text" name = "any name” size = "30" value = " " maxlength = " " >**

**Password input** element is exactly the same as of the text input element. The difference is that in password input element the data is written in encrypted form and we cannot see the data of the viewer. The data is decrypted when it is received on the server side. The html and attributes of password input element are:

**< input type = "password" name = "any name” size = "30" value = " " maxlength = " " >**

**Text area** is multi-line field and can scroll as the viewer enters more text. The html and attributes of text area input element are:

**< text area name = "any name " rows = "N" cols = "N" wrap = "virtual" > < /textarea >**

Where "N" is any numerical value and rows and columns identify size of the text area. Wrap equals virtual, the text entered will automatically wrap at the right side of the field.

**Radio button** is used when user wants to select one choice from among many. While declaring radio buttons name must be same as you will see in the attributes below.

**< input type = "radio" name = "any name" value = " any value" checked>**

The "checked" option will make that button highlighted when the page loads.

**Checkboxes** are similar to the radio buttons but here we can select many choices instead of one and unlike radio buttons every checkbox must has a unique name.

**< input type = "checkbox" name = "any name" value = " any value" checked>**

The "checked" option will make that checkbox highlighted when the page loads.

**Submit button** is used to create buttons on the form. The submit button tells the browser to gather all the selections, values and entered text in the form elements and sends it off to the place defined in the form tag of the server. The html and attributes of submit button are:

**< input type = "submit" value = "send this data now" >**

The value option defines the text that will display on the button.

**Reset button** is used to restore the form to its default state that how it looked when the viewer first entered the page. The html and attributes of reset button are:

**< input type = "reset" value = "clear this web form" >**

The value option defines the text that will display on the button.

**HTML5** is a core technology markup language of the Internet used for structuring and presenting content for the World Wide Web. It is the fifth revision of the HTML standard. Its core aims have been to improve the language with support for the latest multimedia while keeping it easily readable by humans and consistently understood by computers and devices. HTML5 is a response to the fact that the HTML and XHTML in common use on the World Wide Web are a mixture of features introduced by various specifications, along with those introduced by software products such as web browsers, those established by common practice, and the many syntax errors in existing web documents. It is also an attempt to define a single markup language that can be written in either HTML or XHTML syntax. It includes detailed processing models to encourage more interoperable implementations; it extends, improves and rationalizes the markup available for documents, and introduces markup and application programming interfaces (APIs) for complex web applications. For the same reasons, HTML5 is also a potential candidate

for cross-platform mobile applications. Many features of HTML5 have been built with the consideration of being able to run on low-powered devices such as smartphones and tablets.

HTML5 adds many new syntactic features. These include the new <video>, <audio> and <canvas> elements, as well as the integration of scalable vector graphics (SVG) content (replacing generic <object> tags), and MathML for mathematical formulas. These features are designed to make it easy to include and handle multimedia and graphical content on the web without having to resort to proprietary plugins and APIs. Other new elements, such as <section>, <article>, <header> and <nav>, are designed to enrich the semantic content of documents. New attributes have been introduced for the same purpose, while some element and attributes have been removed. Some elements, such as <a>, <cite> and <menu> have been changed, redefined or standardized. The APIs and Document Object Model (DOM) are no longer afterthoughts, but are fundamental parts of the HTML5 specification.

**Step 1:** Open the Notepad, write the following HTML code and save file as Lab01.html

```
<html>
<head><title>Web Engineering Lab 01</title></head>
<body>
<h2>Sign Up Form</h2>
<form action="" method="post" >
<table>
<tr>
 <td width="10%">Name:</td>
 <td width="90%"><input type="text" name="name" ></td>
</tr>
<tr>
 <td>Gender:</td>
  <td><input type="radio" name="gender" value="male" checked >Male <br>
  <input type="radio" name="gender" value="female" >Female</td>
</tr>

<tr>
  <td>Email:</td>
  <td><input type="text" name="email" ></td>
</tr>

<tr>
  <td>Address:</td>
  <td><textarea ></textarea></td>
</tr>

<tr>
```

```
    <td>Login ID:</td>
    <td><input type="text" name="login" ></td>
  </tr>

  <tr>
    <td>Password:</td>
    <td><input type="password" name="pass" ></td>
  </tr>

  <tr>
    <td>Agreed with<br> Policy:</td>
    <td><input type="checkbox" name="agree" ></td>
  </tr>

  <tr>
    <td></td>
    <td ><input type="submit" value="Submit">   <input type="reset"
    value="Reset"></td>
  </tr>
</table>
</form>
</body>
</html>
```

**Step 2:** Open the Lab01.html in web browser


Figure 1: Signup page in HTML

**Step 3:** Open the Notepad, write the following HTML 5 code and save file as Lab01a.html.

```
<!DOCTYPE html>
<html>
<body>

 <form action="#">

  HTML 5 color Element <br>
  Select your favorite color: <input type="color" name="favcolor"><br>
   <input type="submit"><br /><br />

   HTML 5 input restriction as number<br>
   Quantity (between 1 and 5):
   <input type="number" name="quantity" min="1" max="5">
   <input type="submit" value="Send">
   <br><br>

 HTML 5 data Encryption<br>
 Username: <input type="text" name="usr_name">
 Encryption: <keygen name="security">
        <input type="submit"><br><br>

  HTML 5 date picker <br>
  Birthday:
  <input type="date" name="bday">
  <input type="submit" value="Send">

 </form><br><br>

 HTML 5 Audio Element <br>
 <audio controls>
 <source src="horse.ogg" type="audio/ogg">
 <source src="horse.mp3" type="audio/mpeg">
 </audio><br>
 <br>

 HTML 5 CANVAS (graphics and text)<br>
 <canvas id="myCanvas" width="200" height="100" style="border:1px solid #000000;">
 </canvas><br><br>
 <script>

   var c = document.getElementById("myCanvas");
         var ctx = c.getContext("2d");
         ctx.font = "30px Arial";
         ctx.fillText("Hello World",10,50);
 </script><br><br>

</body>
```

```
</html>
```

**Step 4:** Double click on Lab01a.html file and file will open in default browser viewing following output as shown in figures below

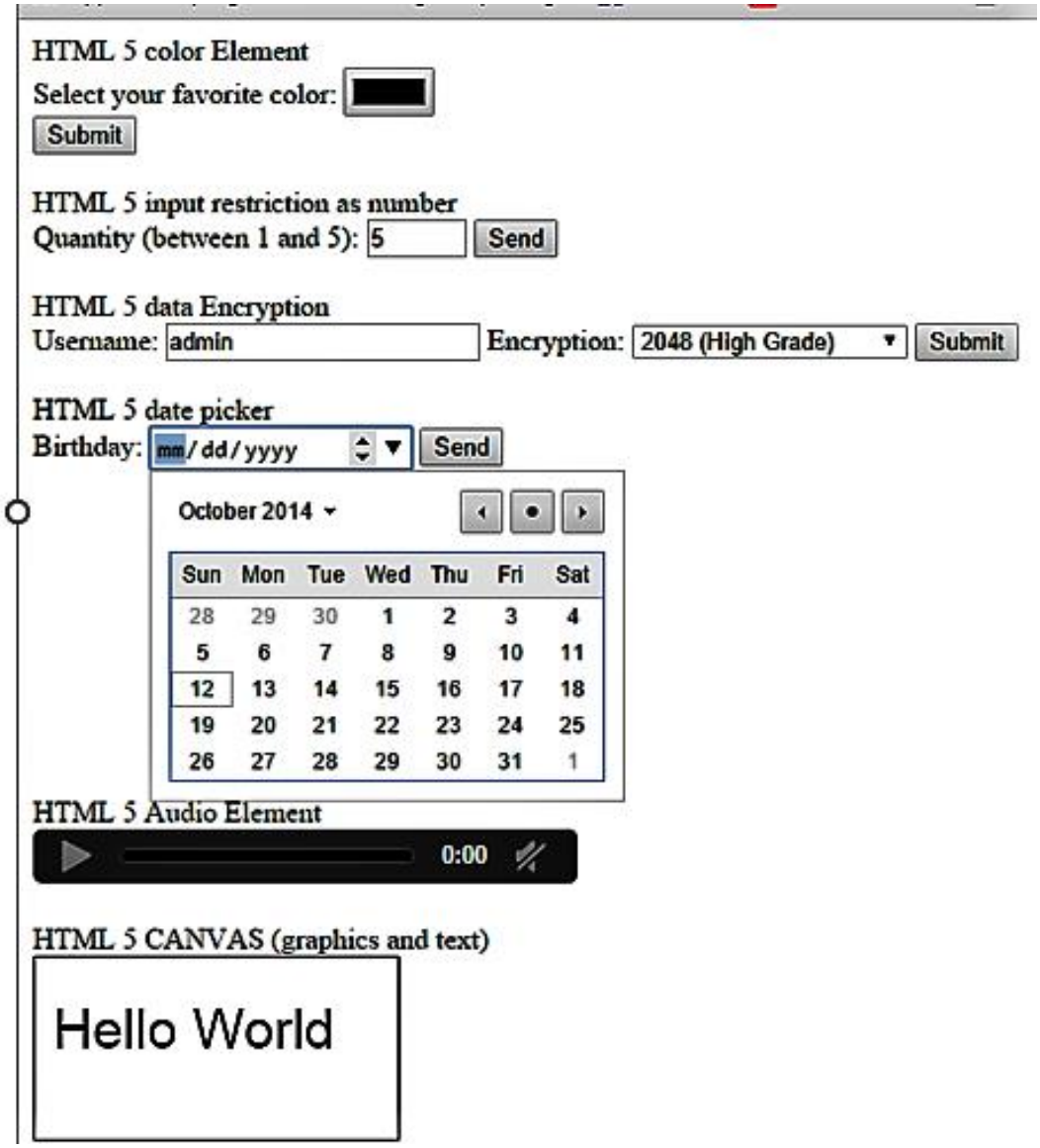



Figure 1 (a): HTML 5 elements output

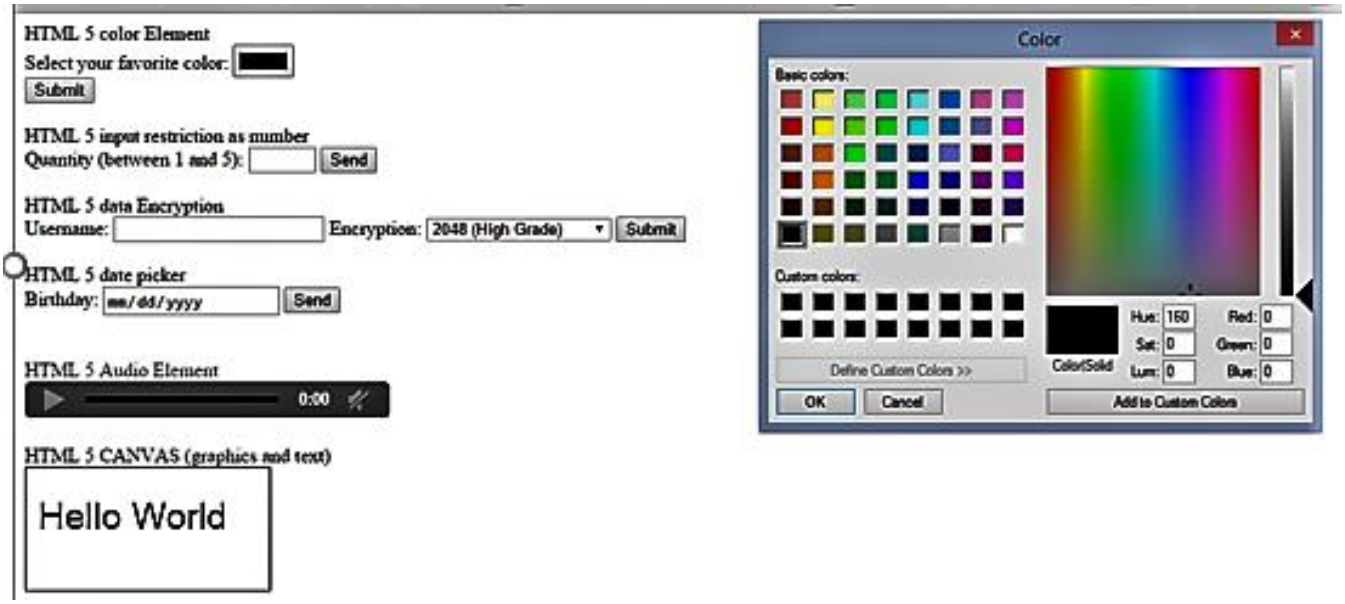



Figure 1 (b): HTML 5 elements

**Lab task(s):**

1. Create following web page using ordered lists, unordered lists and definition lists.

1. Chapter 1
    A. Introduction
    B. HTML
2. Chapter 2
    I. Tags
    II. Lists
        - This is first item of the second item
            1. And this is a numbered subitem of first subitem
        - This is second item of the second item

Title 1
: Definition 1

Title 2
: Definition 2
: Definition 3
    - Unordered list item 1
    - Unordered list item 2

3. Chapter 3

Figure 2: lists

2. Create following registration form using HTML Form controls and tables.

# REGISTRATION FORM

| INFORMATION TABLE | |
|---|---|
| NAME | ENTER YOUR FULL NAME |
| EMAIL | ENTER AN INTERNET CONTACT ADDRESS |
| PASSWORD | ENTER A CODE TO IDENTIFY YOU |
| ADDRESS | ENTER COMPLETE ADDRESS |
| COUNTRY | country |
| GENDER | Male Female |
| HOBBIES | Reading<br>Listening Music<br>Gardening<br>Swimming<br>Athletics<br>check all that apply |
| YOUR WEBSITE | http://<br>provide your website URL if exists (not necessary) |
| OTHER INFORMATION ABOUT YOU | write any other information that is not present here |
| SEND DETAILS | |
| | SEND RESET |

*a simple form*
*designed and developed by* ***Just Bring*** *contact at* ***justbring1123@gmail.com***


Figure 3: Registration Form

3. Use HTML 5 tags only to implement the following in Figure 04.

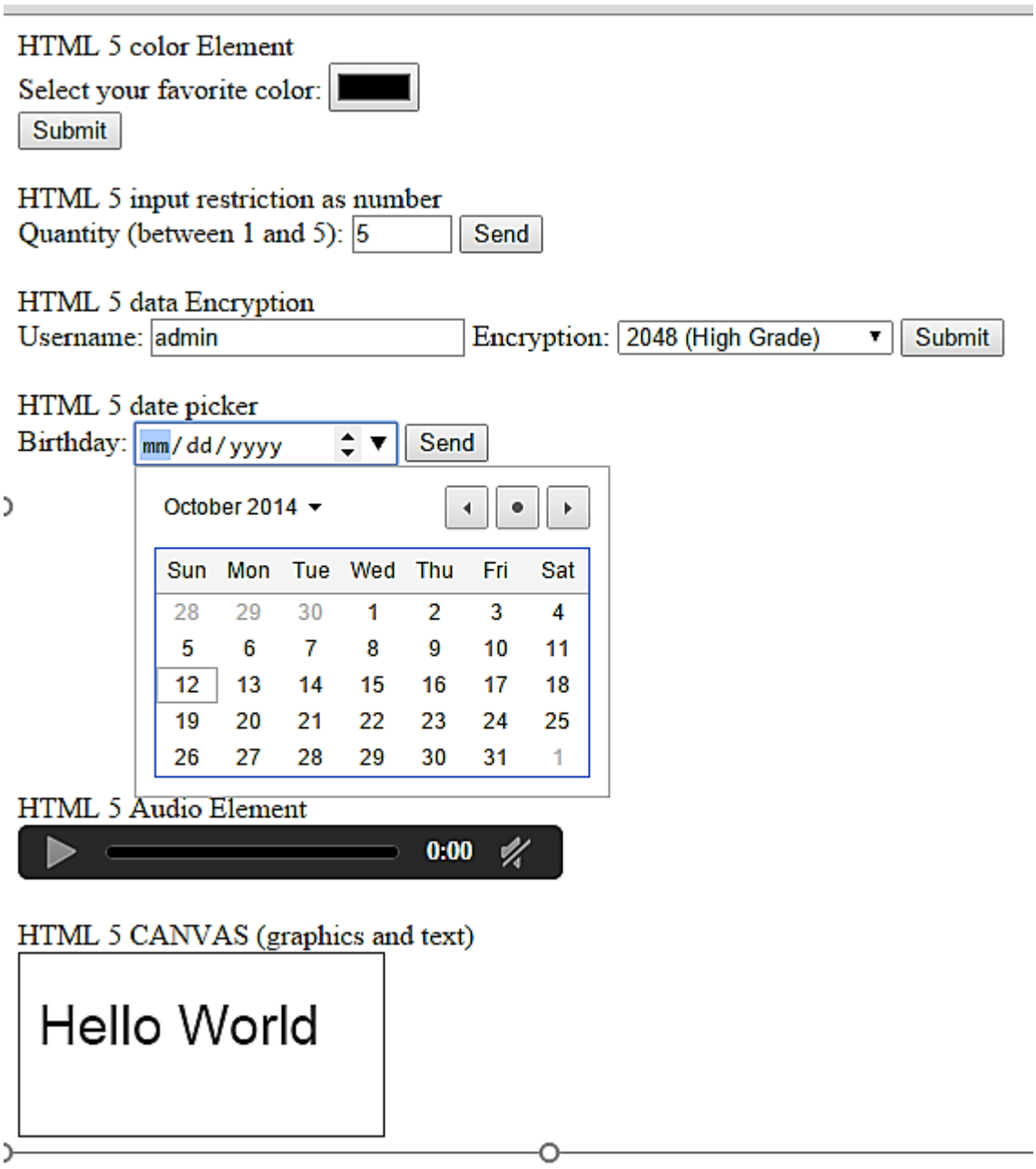



Figure 4: HTML 5 Elements

## Experiment No. 2 Implementation of CSS types, selectors, box-model

**Objectives:** To familiarize students with different CSS styles, selectors, box-model, borders, outline, Classes and IDs.

**Tools:** Dreamweaver, Browser (Internet Explorer, Google Chrome or Firefox)

**Procedure:** Creating an html file by applying different CSS styles.

**Cascading Style Sheets (CSS)** is a style sheet language used for describing the look and formatting of a document written in a markup language. CSS is designed primarily to enable the separation of document content from document presentation, including elements such as the layout, colors, and fonts. This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple HTML pages to share formatting by specifying the relevant CSS in a separate ***.css file***, and reduce complexity and repetition in the structural content.

CSS files can be embedded in HTML in three different categories such as external (a separate style sheet is created which contains all CSS code, save with extension of .css and is embedded in HTML page via ***link tag***), internal (use CSS code in HTML **style** tag) or inline (using CSS code within HTML elements tag).

A style sheet contains a list of rules having selectors and a declaration box such as:

***Selector***

***{***

**CSS statements;**

**}**

In CSS, **selectors** are used to declare which part of the markup a style applies to by matching tags and attributes in the markup itself.

A **declaration block** consists of a list of declarations in braces. Each declaration itself consists of a property, a colon (:), and a value. If there are multiple declarations in a block, a semi-colon (;) must be inserted to separate each declaration.

**Classes and IDs** are case-sensitive, start with letters, and can include alphanumeric characters and underscores. Any number of instances of any number of elements may have the same class. Conventionally, IDs only apply to one instance of one element.

**Step 1:** Open Dreamweaver, create CSS file from File tab, write the following CSS code and save file as Lab2.css

```
p{
        border:groove;
        border-color:#4f8;
        border-radiud:20px;
```

```
        padding-top:5px;
        padding-left:15px;
        margin-bottom:18px;
        text-align:center;
}
img{
        height:300px;
        width:500px;
        transform: rotate(10deg);
        margin-top:20px;
}
```

**Step 2:** Create HTML file from file tab in Dreamweaver, write the following HTML and CSS code and save as Lab02.html

```
<html>
<head><title>Web Engineering Lab 02</title></head>
<link rel="stylesheet" href="lab2.css"  />
<style>
h{
font-style:italic;
float:left;
}
h2{
color:green;
}
div{
margin-left:60px;
background-color:aqua;
}
</style>
<body>
<h1>heading is italic and floating left by using embedded style.</h1>
<h2>getting style from h style and also h2 style.</h2>
<div> hello world ,its embeded css </div>
<p> This styling comes from external sheet. </p>
<img src="jameshirin.jpg"/>
the image is styled from external sheet.
</body>
</html>
```

**Step 3:** Open the Lab02.html page in web browser

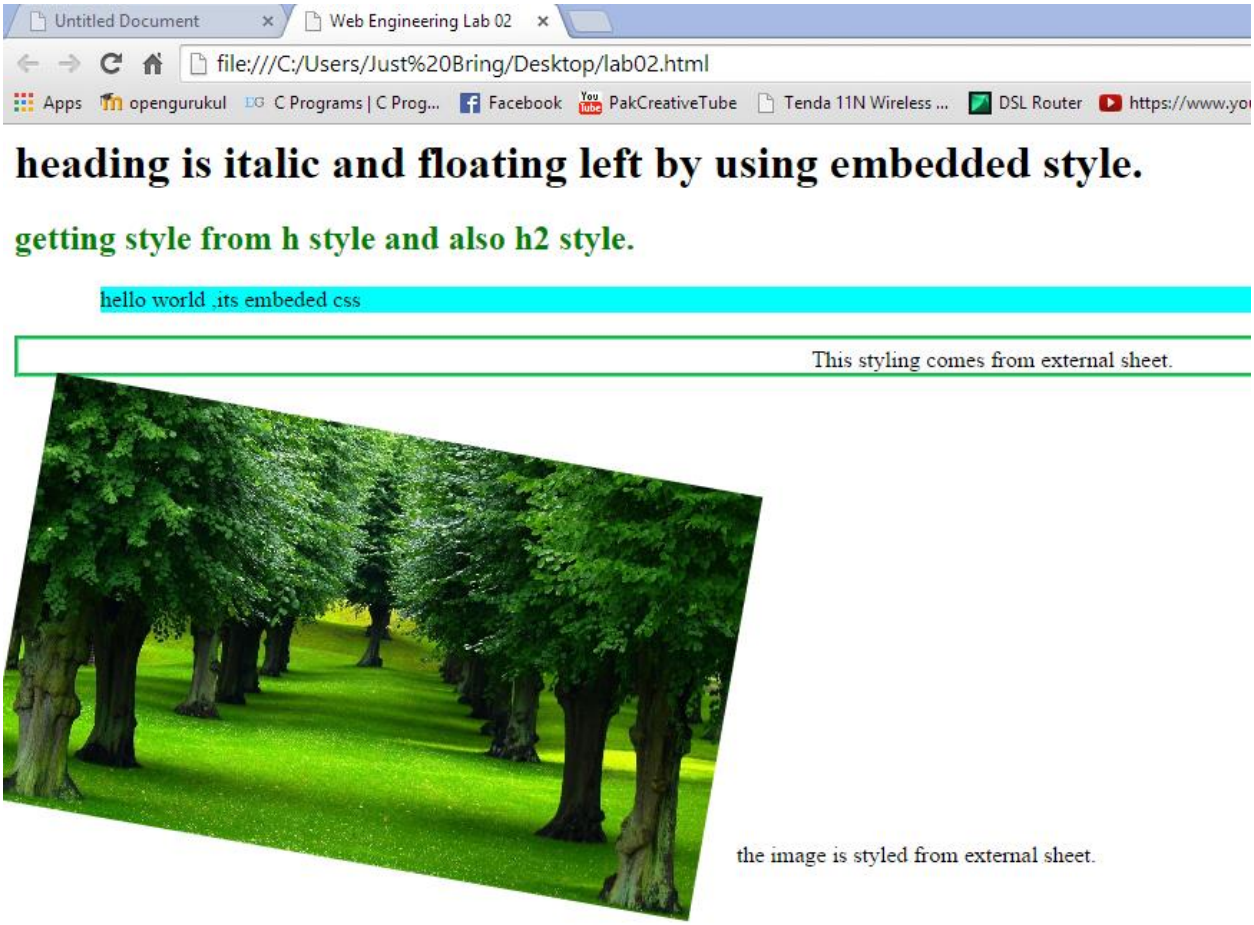



Figure 5: Demonstration of CSS Styles

The **Box Model** in CSS is a fundamental concept that defines how elements on a web page are structured and how their sizes and positions are determined. The box model consists of four main parts:

1. **Content**: The actual content of the element, such as text or an image.
2. **Padding**: The space between the content and the border. Padding is transparent.
3. **Border**: A line that surrounds the padding (if any) and content.
4. **Margin**: The space outside the border, which separates the element from its neighbors. Margins are also transparent.

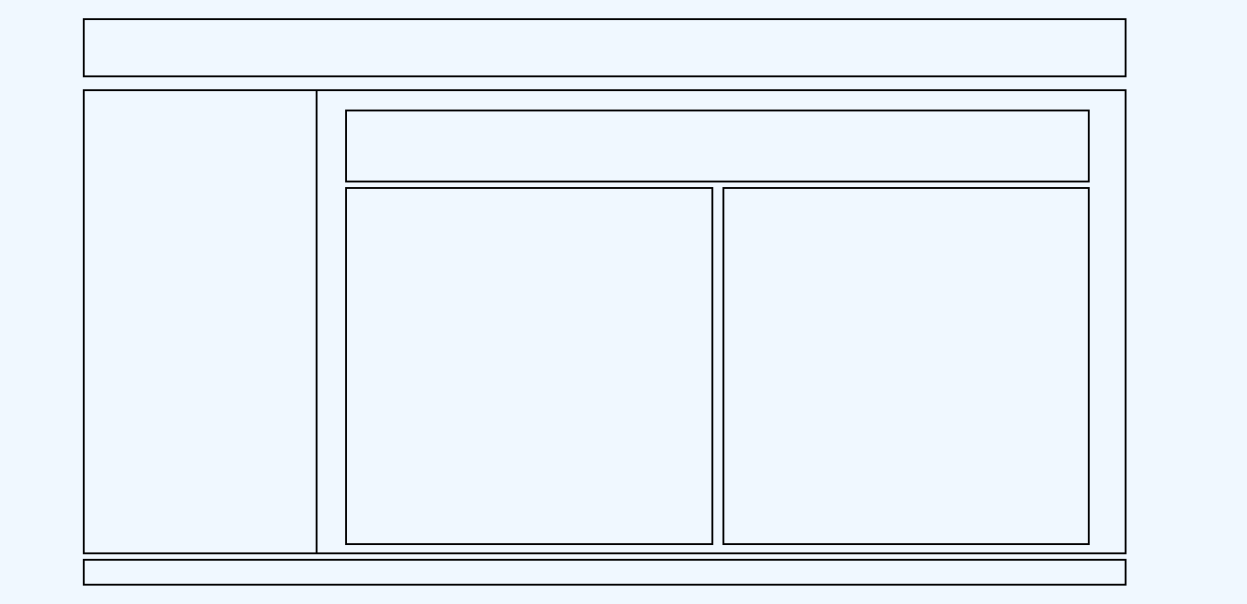
These parts work together to create the total size of an element. The width and height of an element are determined by the content, padding, border, and margin.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <style>
    .box {
```

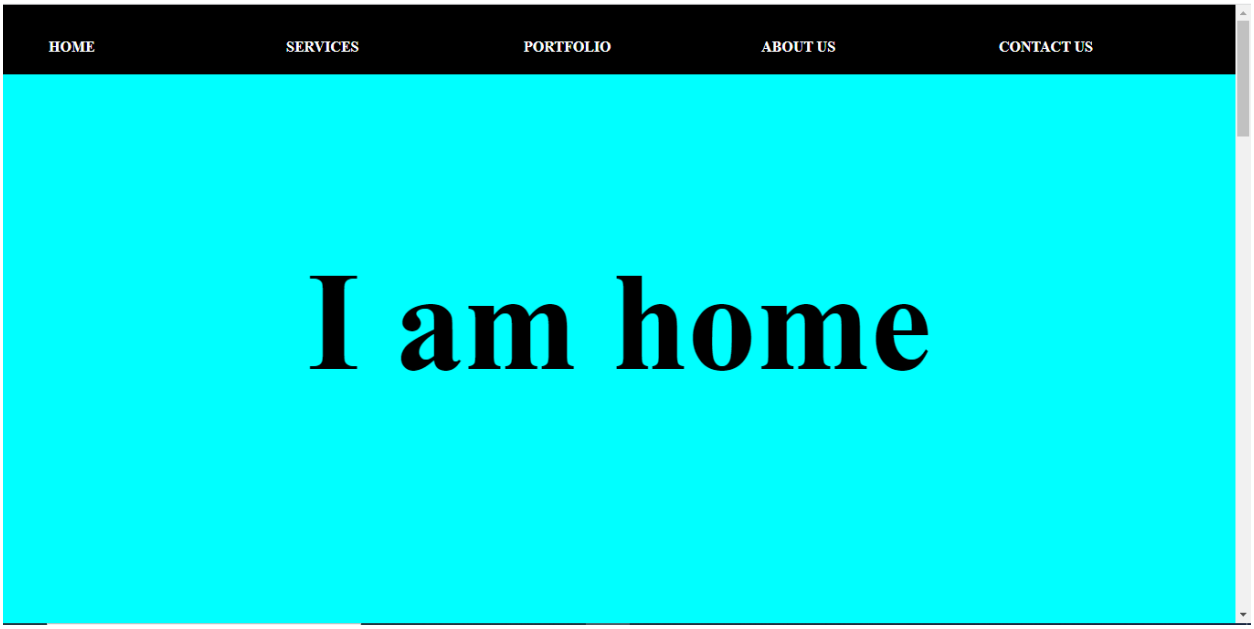
```
        width: 200px; /* Content width */
        height: 100px; /* Content height */
        padding: 20px; /* Padding on all sides */
        border: 5px solid black; /* Border width and style */
        margin: 10px; /* Margin on all sides */
        background-color: lightblue;
      }
    </style>
</head>
<body>
    <div class="box">This is a box model example.</div>
</body>
</html>
```

**Question: Write down HTML & CSS code for the following.**

1. Create following website pattern

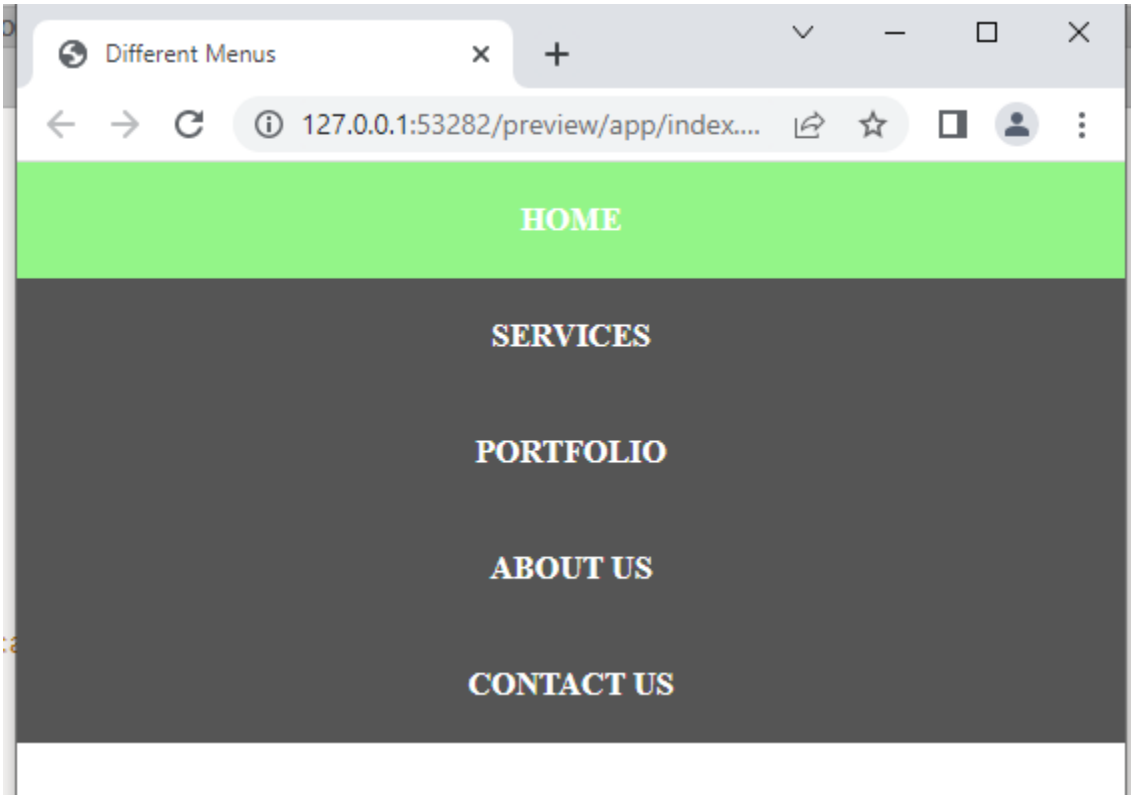


2. Create following one-page website




3. Create menu as follow: The first menu is simple menu with 5 tabs. The second menu should change font size on hovering The third menu should change background color on hovering and must have border as shown in the following figures. The fourth menu should have active table while other menu tabs should change color only.


The fifth menu should be responsive in nature as shown below


Different Menus
127.0.0.1:53282/preview/app/index....
HOME
SERVICES
PORTFOLIO
ABOUT US
CONTACT US

## Experiment No. 3 Implementation of tables, forms, dropdowns, buttons, flexbox

**Objectives:** To familiarize students with different CSS tables, forms, dropdowns, buttons, flexbox and grid.

**Tools:** Dreamweaver, Browser (Internet Explorer, Google Chrome or Firefox)

**Procedure:** Creating an html file by applying different CSS styles.

**1.** Tables

Tables in HTML are used to display data in a tabular format, with rows and columns. CSS can be used to style tables for a more attractive and readable presentation.

```
  <style>
    table {
      width: 100%;
      border-collapse: collapse;
    }
    th, td {
      padding: 10px;
      border: 1px solid #ccc;
      text-align: left;
    }
    th {
      background-color: #f2f2f2;
    }
  </style>
<body>
  <table>
    <thead>
      <tr>
        <th>Name</th>
        <th>Age</th>
        <th>City</th>
      </tr>
    </thead>
    <tbody>
      <tr>
        <td>John Doe</td>
        <td>28</td>
        <td>New York</td>
      </tr>
      <tr>
        <td>Jane Smith</td>
        <td>34</td>
```

```
            <td>Los Angeles</td>
        </tr>
        <tr>
            <td>Emily Johnson</td>
            <td>22</td>
            <td>Chicago</td>
        </tr>
    </tbody>
  </table>
</body>
</html>
```

**2. Forms**

Forms in HTML are used to collect user input. They can contain various input elements such as text fields, checkboxes, radio buttons, and more.

```
  <style>
    form {
      max-width: 400px;
      margin: 0 auto;
      padding: 20px;
      border: 1px solid #ccc;
      border-radius: 10px;
      background-color: #f9f9f9;
    }
    label, input {
      display: block;
      width: 100%;
      margin-bottom: 10px;
    }
    input[type="submit"] {
      width: auto;
      background-color: #4CAF50;
      color: white;
      padding: 10px 20px;
      border: none;
      border-radius: 5px;
      cursor: pointer;
    }
  </style>
<body>
  <form>
    <label for="name">Name:</label>
    <input type="text" id="name" name="name" required>

    <label for="email">Email:</label>
```

```
    <input type="email" id="email" name="email" required>

    <label for="password">Password:</label>
    <input type="password" id="password" name="password" required>

    <input type="submit" value="Submit">
  </form>
</body></html>
```

**3. Dropdown**

A dropdown menu allows the user to choose one value from a predefined list of options. This is typically implemented using the <select> tag.

```
  <style>
    select {
      padding: 10px;
      width: 100%;
      border-radius: 5px;
      border: 1px solid #ccc;
      background-color: #f9f9f9;
    }
  </style>
</head>
<body>
  <label for="city">Choose a city:</label>
  <select id="city" name="city">
    <option value="newyork">New York</option>
    <option value="losangeles">Los Angeles</option>
    <option value="chicago">Chicago</option>
    <option value="houston">Houston</option>
  </select>
</body>
</html>
```

**4. buttons**

Buttons are interactive elements that users can click to perform actions, such as submitting a form or triggering a JavaScript function.

```
<style> .button { background-color: #4CAF50; color: white; padding: 15px 20px; text-align: center; text-decoration: none; display: inline-block; font-size: 16px; margin: 4px 2px; cursor: pointer; border-radius: 8px; border: none; } </style>
</head> <body>
```

```
<button class="button">Click Me!</button>
 </body> </html>
```

**4. flexbox**

Flexbox is a layout model in CSS that allows elements within a container to be distributed and aligned efficiently along a single axis or both axes. It provides an easier and more flexible way to design responsive layouts.

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <style>
      .container {
         display: flex;
         justify-content: space-around;
         align-items: center;
         height: 200px;
         background-color: #f2f2f2;
      }
      .box {
         width: 100px;
         height: 100px;
         background-color: lightblue;
         display: flex;
         justify-content: center;
         align-items: center;
      }
   </style>
</head>
<body>
   <div class="container">
      <div class="box">Box 1</div>
      <div class="box">Box 2</div>
      <div class="box">Box 3</div>
   </div>
</body>
</html>
```

**5. flexbox**

CSS Grid Layout is a powerful layout system for creating complex, responsive layouts. Unlike Flexbox, which is one-dimensional (row or column), Grid allows for two-dimensional layouts (both rows and columns).

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <style>
    .grid-container {
      display: grid;
      grid-template-columns: repeat(3, 1fr);
      gap: 10px;
      background-color: #f9f9f9;
      padding: 10px;
    }
    .grid-item {
      background-color: lightcoral;
      padding: 20px;
      text-align: center;
      border-radius: 5px;
    }
  </style>
</head>
<body>
  <div class="grid-container">
    <div class="grid-item">1</div>
    <div class="grid-item">2</div>
    <div class="grid-item">3</div>
    <div class="grid-item">4</div>
    <div class="grid-item">5</div>
    <div class="grid-item">6</div>
  </div>
</body>
</html>
```

**Lab Tasks: Perform the following:**

1. Create following web page and clearly indicate header, menu, side menu and contents using Cascading Style Sheets and Hyper Text MarkUp Language.
2. The Logo should be as “Coding is Poetry” inline with search box and search button. The search box should have search icon and this icon could be referenced by using any icon library depends upon your choice.

3. The menu should consists of “HOME, WEB DEVELOPMENT, DESKTOP LEARNING, REGISTRATION, CONTACTS”. The first item of the menu “HOME” should be active and all the items should be changed upon hovering of mouse as shown in figure 1 with menu item “DESKTOP LEARNING”.


**Figure 3.1: first part of web page**

4. There should be two side sections of the web page. Both list items should be hovered upon mouse as shown in figure 2 below. The left and right sections have list items as images. The images can be changed and can be used of your own choice.
5. The footer section should contain your own name instead of given in figure 2 below. Use Linear Gradient properties while designing footer.

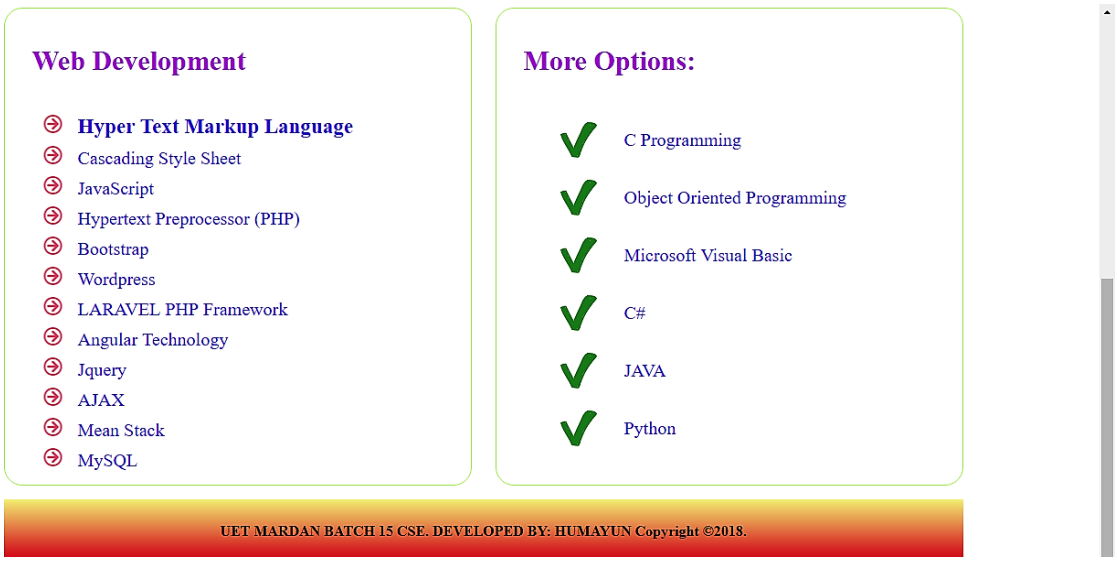



Figure 3.2: Second part of web page


Figure 3: Full web page

## Experiment No. 4 Implementation of JavaScript variables, operators, loop, functions, arrays

**Objectives:** To familiarize students with basic syntax of JavaScript, variables, operators, loop, functions and arrays.

**Tools:** Dreamweaver, Browser (Internet Explorer, Google Chrome or Firefox)

**Procedure:** Prompting the user to input her name and displaying user's name as part of the dialog box message. Also applying JavaScript validation on HTML form by providing required information.

**JavaScript** is a computer language specially designed to work with Internet Browsers. It lets you create small programs called scripts and embed them inside HTML pages to provide interactive content on your web pages. JavaScript is an interpreted language. This means that the script in not compiled before it is executed. We simply write a few lines of code inside an HTML page, save the page, and open the page in your web browser to test how it looks. JavaScript is an object-based scripting language. This means that it views everything as an object. The browser is an object. A window is an object. A button in a window is an object. Every object has properties. And we can use JavaScript to manipulate these properties. JavaScript supports event-driven programming. An event is an action that occurs when the user does something such as click a button or move the pointer over a graphic image.

JavaScript tags are embedded in HTML pages using **<script>…</script>** tags. The text inside script tags does not appears in the browser window—except for older browser that do not understand JavaScript. The tags can be placed within either the haead or body section of an HTML page. JavaScript tags can be referenced as an external .js file. JavaScript can also be placed with the HTML tags.

To use a variable in your script, that variable must first be declared by using *"var"* keyword as demonstrated below:

**var first_name = "humayun";**

We can also declare a variable by simply referencing it for the first time as shown in the following example:

**first_name = "humayun";**

Variable names can consist only of uppercase and lowercase letters, the underscore character, and the numbers 0 through 9 and is case sensitive too. Variable names cannot contain spaces and cannot be used as reserved words. Scope of variable can be as local (that is explicitly declared using the "var" statement insides a function.) or global (any variable defined outside of a function).

For displaying output in a web browser, the following syntax is used:

**document.write("String");**

**Functions –** a function is a collection of statements that perform a given task. Most functions are defined in the head section of the HTML page because we will always know where our functions are and second reason is that the head section always executes first, ensuring that functions are available when needed later in script. Function can be called by its name and ending with a semicolon as *FunctionName();*. Syntax for a function is shown here:

```
function FunctionName (arguments)
{
    Statements;
}
```

**Array –** an indexed list of values that can be referred to as a unit. Arrays can contains any type of JavaScript value. Before using an array, it must be declare or define. Array values can be accessed via its index number starting from digit 0. The following shows how to create and array that will hold string values.

```
Auto      = new Array (2);
Auto [0]  = "humayun";
Auto [1]  = "khan";
```

***New*** keyword is used to create an Array object named **Auto** that will contain 2 entries. The rest of the statements assign string values to the array. Array contents can be display via loops.

**Lab Task: Write down HTML, CSS & JavaScript code for the following.**

1. Implement the concept as how to place JavaScript in the body section, head section and referencing JavaScript as an external file.
2. Create a function that helps in the addition of two integers.
3. Produce the following

```
x  =  y + 1  ...Result = 3
x  +=  y  ...Result = 6
x  -+ y  ...Result = 3
x  *= y  ...Result = 9
x  /= y  ...Result = 3
```

4. Prompt user to input 3 integers and return its sum and average in a dialogue box message.

5. Create a dropdown list of your own choice but the elements/data of the dropdown should come through JavaScript array concept not via HTML traditional concept.
6. Implement student grade sheet where the user should enter registration number, name, semester, subject marks such as Web Engineering, computer programming, object oriented programming, Data Structures and Algorithms and Artificial intelligence. The system should calculate the percentage of the student based on 500 marks and return the grades such as A, B, C, D or F. Set your own grading criteria. The output should be displayed in tabular format depicting the registration number, name, individual subject marks, obtained marks and final grade.
7. Implement a standard calculator having addition, subtraction, multiplication and division functionalities only. The script should call an external scripting file, prompt user for the first operand, followed by any operator and second operand. Based on the operator provided by the user, the system should call the corresponding function and alert the desired result. Do not use the conditional if-else statement.

## Experiment No. 5 Implementation of JS objects, Events, Form Validation

**Objectives:** To familiarize students with JavaScript objects, form validation and event handling.

**Tools:** Dreamweaver, Browser (Internet Explorer, Google Chrome or Firefox)

**Procedure:** Prompting the user to input her name and displaying user's name as part of the dialog box message. Also applying JavaScript validation on HTML form by providing required information.

**Form validation –** browser created a form object for every form element defined by <FORM> tags. We can assign a name to each element using the optional NAME = "" attribute for each element, or can use the forms [] array, which contains an index listing of every form element on the page beginning with a value of index of 0. Each form contains its own elements [] array that contains an aentry for each form elements on that particular form. For example, the fourth form element on a form called myForm can be reference as myForm.elements [3].

The real benefit of using JavaScript with your forms is to perform validation of use input. Validation allows you to ensure that the user has filled in all required fields and that valid data has been entered in those fields. When the use makes a mistake, you can display alert messages explaining the error and asking the user to correct the problem before submitting the form again.

**JavaScript Events –** an event occurs within the confines of the browser. Events include such activities as mouse clicks, mouse movement, pressing keyboard keys, the opening and closing of windows and frames and the resizing of windows. Each event is associate d with a specific object. When an event occurs for a given object, its event handler (a trap that recognizes the occurrence of a particular type of event) executes. For example, the click event occurs whenever a user clicks on a button, document, check box, link, radio option, reset button, or submit button.

**JavaScript Animations -** means number of DOM elements (<img/>,  or otherwise) are moved around the page according to some sort of pattern determined by a logical equation or function. JavaScript animations needed when we need significant control over animations like dynamically track a touch position, or an animation that we need to pause, stop or slow-down or if we are interested in making buttons light up or making browser pages come alive with movements. To achieve the effect of animation, elements must be moved at a given interval or frame rate; from a programming perspective, the simplest way to do this is to set up an animation loop with a delay.

**Step 1:** Open Dreamweaver, write the following HTML and JavaScript code and save file as Lab03.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
```

```
<head>
	<script>
	<!-- Javascript function -->
		function sayHello(visitor_name)
		{
			window.alert("Hello and welcome, " + visitor_name);
		}
	</script>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Web Engineering Lab 3</title>
</head>

<body>
	<script> <!-- Start of Script -->

		name = window.prompt("What is your name?","");
		sayHello(name);

	</script> <!-- End of Script -->

</body>
</html>
```

**Step 2:** Open the Lab03.html in web browser

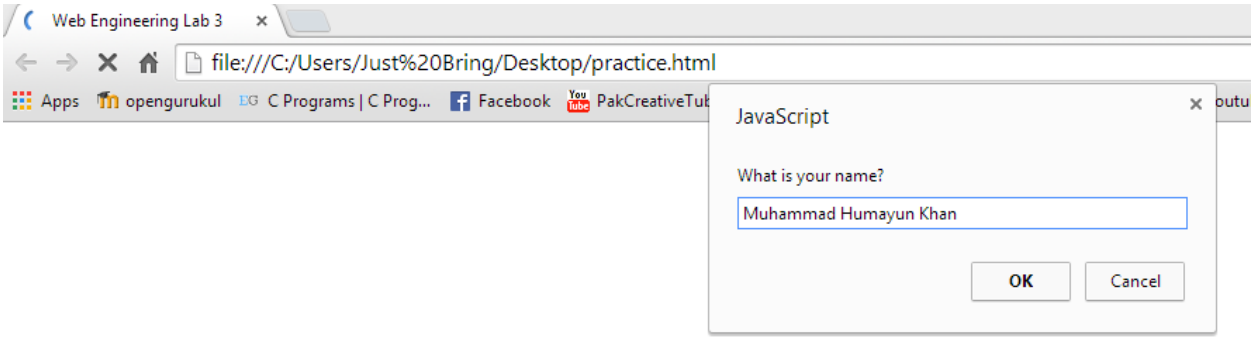



Figure 5.1: Prompt user's input

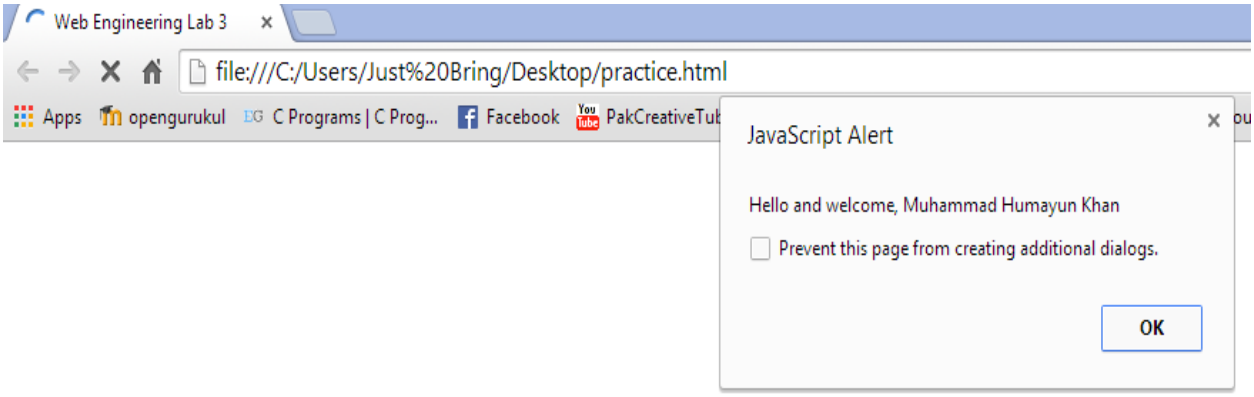



Figure 5.2: Display user's input

**Step 3:** Write the following HTML and JavaScript code and save file as Lab03a.html

```
<html>
<head><title>Web Engineering Lab 03</title></head>
<script language="javascript">
function validation(){
        if(document.getElementById("name").value == ''){
        alert("Please Enter Name");
        return false;
        }
        else if(document.getElementById("email").value == ''){
        alert("Please Enter Email");
        return false;
        }
       else if(document.getElementById("add").value == ''){
        alert("Please Enter Adrees");
        return false;
        }
       else if(document.getElementById("login").value == ''){
        alert("Please Enter Login ID");
        return false;
        }
         else if(document.getElementById("pass").value == ''){
        alert("Please Enter Password");
        return false;
        }
       else if(!document.getElementById("chk").checked){
```

```
        alert("Please Agree with Policy");
        return false;
        }
}
</script>
<body>
<h2>Sign Up Form (Java Script Validation)</h2>
<form action="" method="post" onSubmit="return validation();" >
<table>
<tr>
<td width="10%">Name:</td>
<td width="90%"><input type="text" name="name" id="name" ></td>
</tr>
<tr>
<td>Gender:</td>
<td><input type="radio" name="gender" value="male" checked >Male <br>
<input type="radio" name="gender" value="female" >Female </td>
</tr>
<tr>
<td>Email:</td>
<td><input type="text" name="email" id="email" ></td>
</tr>
<tr>
<td>Address:</td>
<td><textarea name="add" id="add" ></textarea></td>
</tr>
<tr>
<td>Login ID:</td>
<td><input type="text" name="login" id="login" ></td>
</tr>
<tr>
<td>Password:</td>
<td><input type="password" name="pass" id="pass" ></td>
</tr>
<tr>
<td>Agreed with<br> Policy:</td>
<td><input type="checkbox" name="agree" id="chk" ></td>
</tr>
<tr>
<td></td>
```

```
<td ><input type="submit" value="Submit">   <input type="reset"
value="Reset"></td>
</tr>
</table>
</form>
</body>
</html>
```

**Step 4:** Open the Lab03a.html in web browser

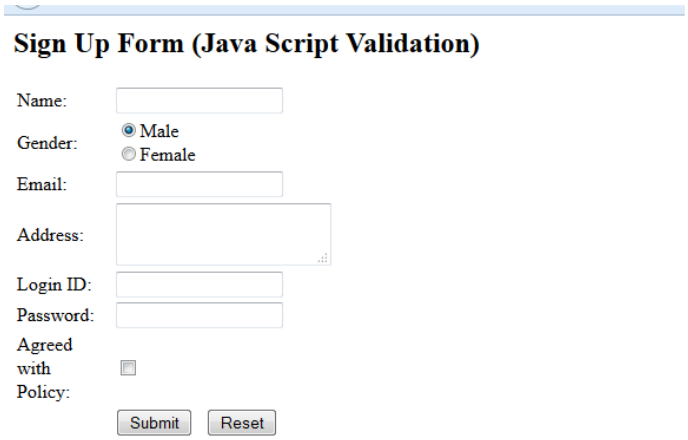



Figure 5.3: Empty Sign up Form

**Step 5:** Fill the form and keep any field invalid like don't agree with policy and press the submit button to check the JavaScript validation as shown in figure below.

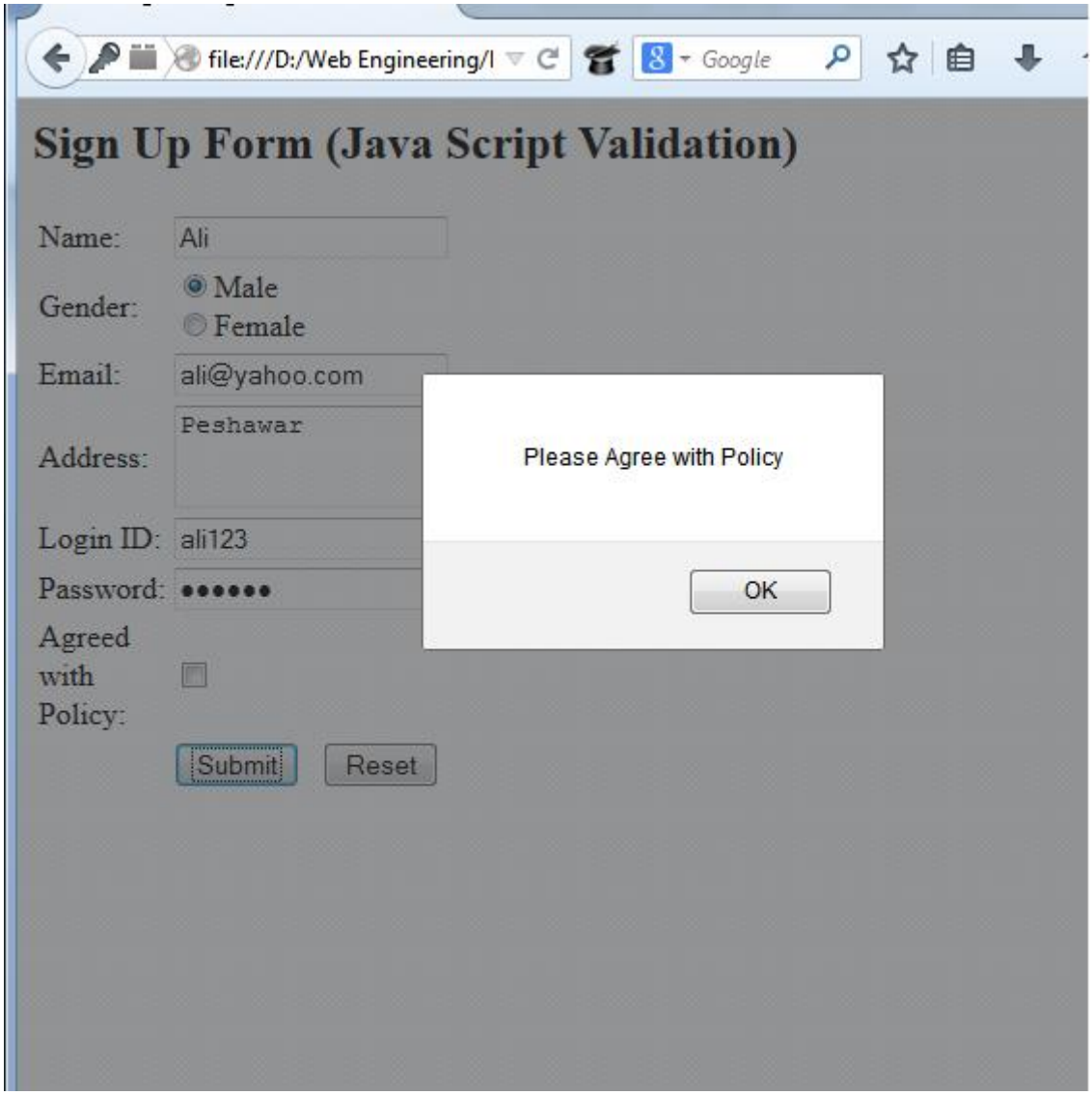



Figure 5.6: Sign up Form Validation

**Step 5:** Create New HTML file in Dreamweaver, write the following HTML and JavaScript code and save file as Lab03b.html

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>JavaScript Events</title>
</head>
<body>
        <h3 id="heading">Introduction to JavaScript Events</h3>

<!--    Single, and double click events -->
```

```
        <form>
        <input type="button" value="Click Me" onClick="events();">
        <input type="button" value="Double Click" onDblClick="events_Dbl();" title="Heading
text changed.">
        <input type="button" value="Double Click" onDblClick="events_Dbl_Two();"
title="Change Color on Double Click">

        Text Box Testing Click Event  <input type="text" placeholder="Click here"
onClick="txtbox_Testing();">
        </form>

<!-- click event upon paragraph -->
        <p id="clickTesting" onClick="para_Testing_Clk();">I am paragraph and user want to
test single click upon me. Let's try!</p>

        <!-- Mouse events as onMouseOver and onMouseOut -->
        <p id="overMouse" onMouseOver="mouseOver();" onMouseOut="mouseOut();">I
am mouse over and mouse out event.</p>

<!--    Mouse events as onMouseDown & onMouseUp -->
        <div>
        <p id="mouseDownUp" onMouseDown="mouseDown();" nMouseUp="mouseUp();">
        I am mouse down and mouse up event.</p>
        </div>
<!-- Mouse event as onMouseMove      -->
        <div onMouseMove="moveMouse();">
                <p>Mouse Move here</p>
                <span id="mouseMoveDiv"></span>
        </div>

<!-- Keyboard event as onkeyup -->
        <div id="demo_keyboard" >
                <form>
        <input type="text"  id="keyUpTextBox" placeholder="Enter text" nKeyUp="keyup();">
                </form>
        </div>

        <p id="input_TextBox_data"></p>

<!--    Keyboard event as onKeyPress -->
        <div id="demo_keyboard_press" >
        <form>
        <input type="text"  id="keyPressTextBox" placeholder="Enter text"
        onKeyUp="keyPress();">
        </form>
        </div>
```

```
        <p id="input_TextBox_keyPress"></p>
<!--    Keyboard event as onKeyDown -->
        <div id="demo_keyboard_down" >
                <form>
<input type="text"  id="keyDownTextBox" placeholder="Enter text" onKeyUp="keyDown();">
                </form>
        </div>

        <p id="input_TextBox_keyDown"></p>

<input type="text" id="colortxtbox" onKeyDown="colorDown();" onKeyUp="colorUp();">
        <br><br>
        <form name="myform">

        Blur Event   <input type="text" id="blurEvent" onBlur="blur_Event();">
                <br /><br />
        Focus Event   <input type="text" id="focusEvent" onFocus="focus_Event();">
                <br /><br />
        On Change Event   <input type="text" id="onchangeEvent"
        onChange="onChangeEvent();">
                <br /><br />

        On Change Event for Drop Down   <select id="dropdown"
        onChange="dropDownChangeEvent();">
                        <option>item 1</option>
                        <option>item 2</option>
                        <option>item 3</option>
                </select>

                <div>
<p id="dropDownPara">I will change color and font size when you made changes in the above
drop down list.</p>
                </div>

        <input type="text" id="selectText" onSelect="selectTextEvent();">

        </form>
                <p id="selectTextarea"></p>

                <!-- onsubmit and onreset events -->
        <form name="myform" onSubmit="form_Submit()" onReset="form_Reset()">
                Name   <input type="text" /><br /><br />
                Gender  <input type="radio" name="gender" />Male
                <input type="radio" name="gender" />Female <br /><br />
                <input type="submit" value="submit">  
                <input type="reset" value="Reset">
        </form>
```

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

**Step 6:** Double click on lab03b.html and required output will be shown in default browser as shown below in figure.

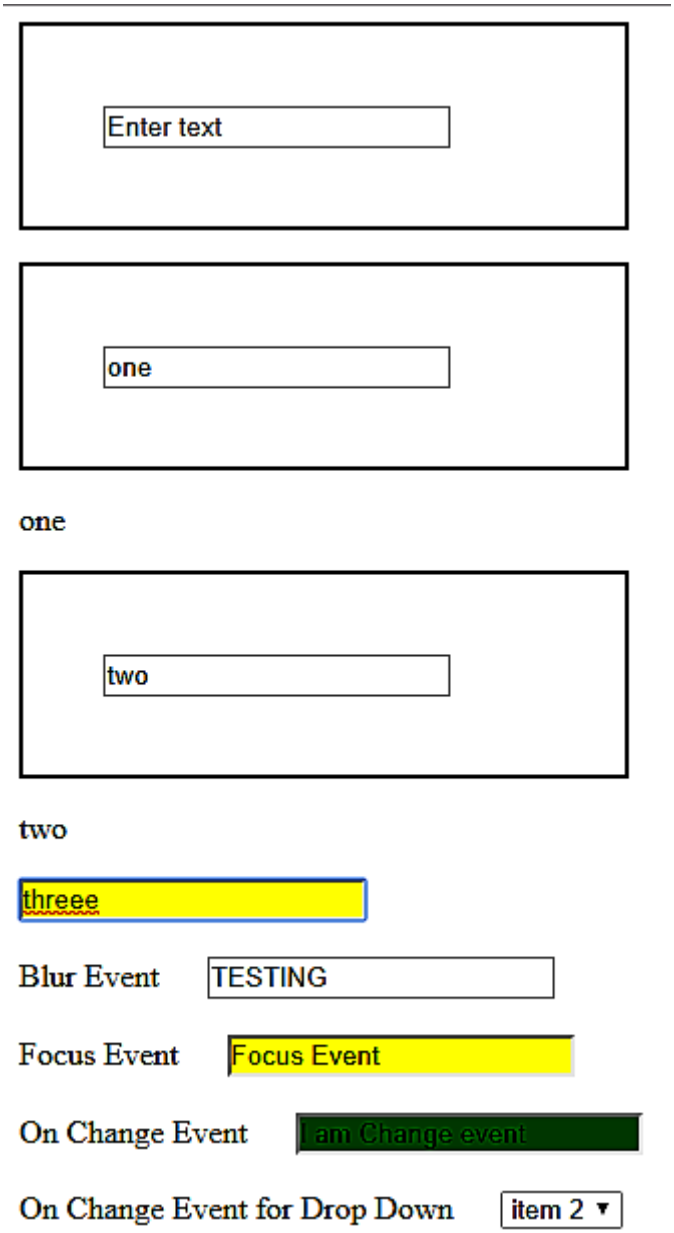
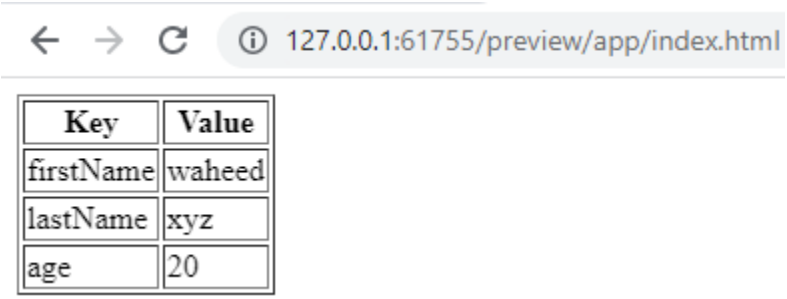


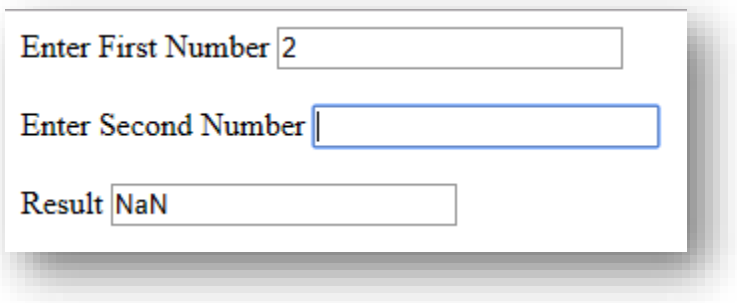

Figure 5.7: JavaScript Events

**Lab Task(s): Perform the following**

1. Create an object having first name, last name and age and display the object key and its associated value via JS concept as shown below

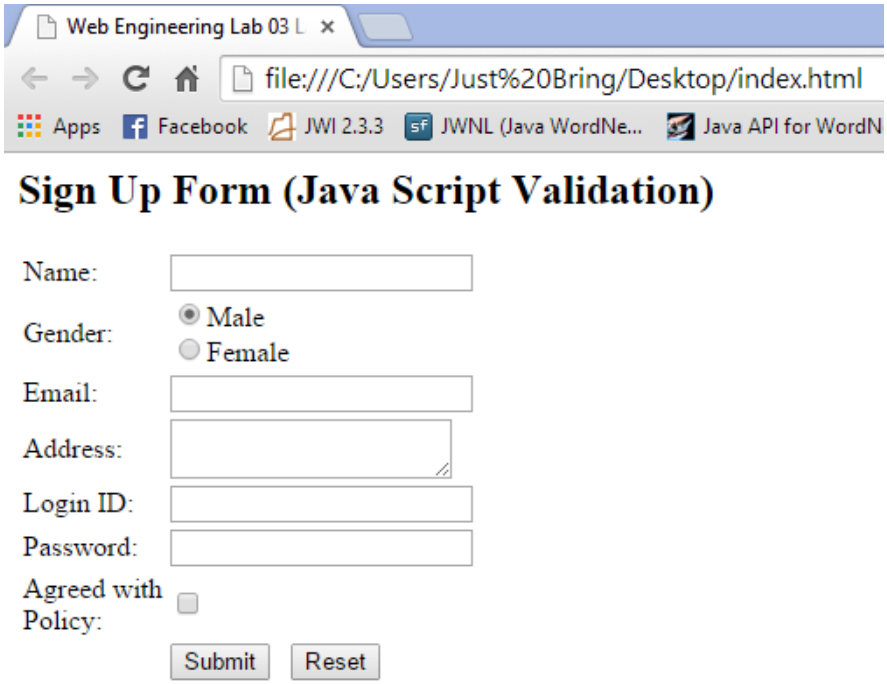


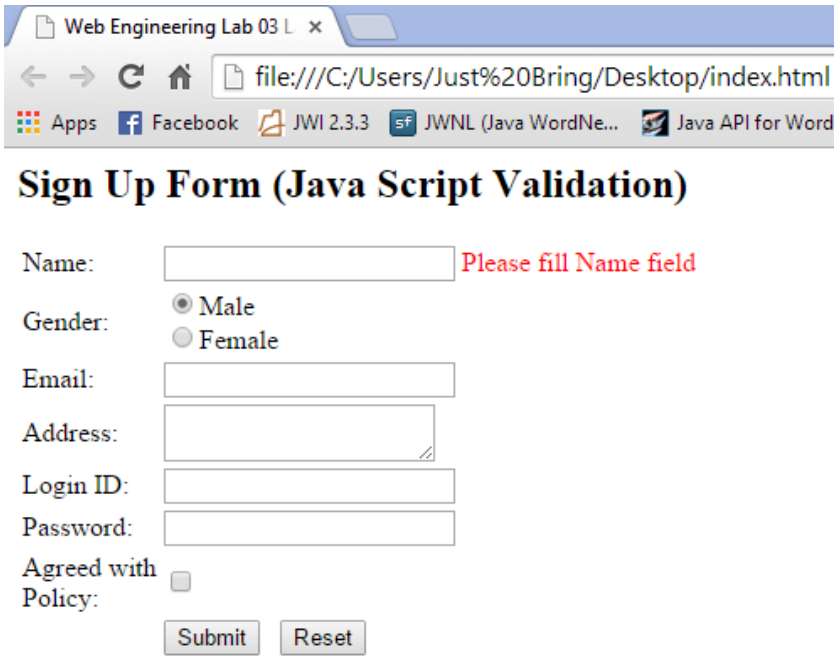

2. Pretend there are three text boxes. The user will provide integer numbers and the addition of two textboxes will be shown in the third textbox. When the user input an integer in the first textbox, the resultant (third) textbox should display answer as NaN. After providing an input into the second textbox the addition result should be displayed in the third textbox. [Hint: Use Event handling concept] as shown below.



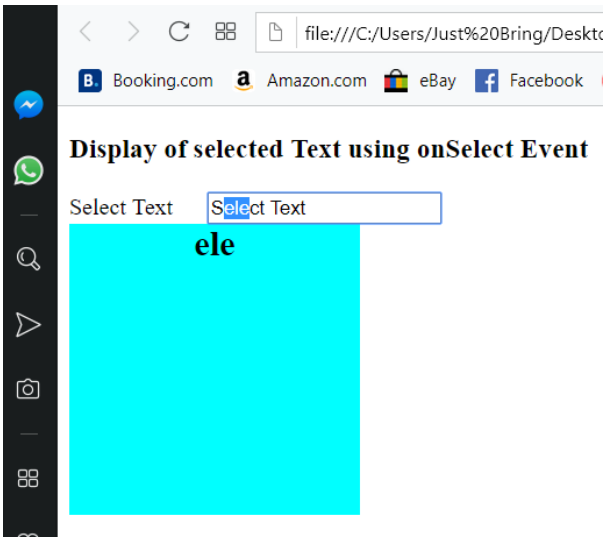

3. Pretend there is a dropdown menu and button against dropdown menu titled as **New Entry.** When the user click the button (**New Entry),** the dropdown menu should be replaced with the textbox as shown below.


4. Apply JavaScript validation for HTML form and show red color message against each invalid input field on form submission as shown below in figures.




5. Implement the following task. Pretend there is a text box in which the user should enter some text. After entering the text, the user should select some or all the text and it should be display in a separate div as show below in the figure.


file:///C:/Users/Just%20Bring/Deskto
Booking.com
Amazon.com
eBay
Facebook
Display of selected Text using onSelect Event
Select Text
Select Text
ele

# Experiment No. 6 Implementation of JS hoisting, arrow function, callbacks, asynchronous, promises, asyn/await

**Objectives:** To familiarize students with JavaScript hoisting, arrow functions, callbacks promises and asynchronous /await

**Tools:** Dreamweaver, Browser (Internet Explorer, Google Chrome or Firefox)

**Procedure:** Executing example for every topic.

1. **Hoisting**

```
console.log(hoistedVar); // Output: undefined
var hoistedVar = 'This variable is hoisted';

hoistedFunction(); // Output: "This function is hoisted"
function hoistedFunction() {
    console.log('This function is hoisted');
}
```

In JavaScript, hoisting refers to the built-in behavior of the language through which declarations of functions, variables, and classes are moved to the top of their scope – all before code execution. In turn, this allows us to use functions, variables, and classes before they are declared.

2. **Arrow Functions**

```
// Traditional function
function add(a, b) {
    return a + b;
}
console.log(add(2, 3)); // Output: 5

// Arrow function
const addArrow = (a, b) => a + b;
console.log(addArrow(2, 3)); // Output: 5
```

In the code above, the addArrow is the variable that holds the function. You can call the function as addArrow ().

(a, b, ...) are the function parameters. You can define as many parameters as required by the function.

Then you have the arrow => to indicate the beginning of the function. After that, you can write curly brackets {} to indicate the function body.

3. **Callbacks**

```
function greet(name, callback) {
    console.log('Hello ' + name);
    callback();
}

function sayGoodbye() {
    console.log('Goodbye');
}

greet('Alice', sayGoodbye); // Output: "Hello Alice" followed by "Goodbye"
```

A **callback function** is a function passed into another function as an argument, which is then invoked inside the outer function to complete some kind of routine or action.

In greet function we passe name and callback as arguments. After doing something inside the greet function we execute the callback() .

4**. Asynchronous Operations**

```
console.log('Start');

setTimeout(() => {
    console.log('This is an asynchronous operation');
}, 2000);

console.log('End');
// Output: "Start", "End", "This is an asynchronous operation"
```

Asynchronous programming is a technique that enables your program to start a potentially long-running task and still be able to be responsive to other events while that task runs, rather than having to wait until that task has finished. Once that task has finished, your program is presented with the result.

Many functions provided by browsers, especially the most interesting ones, can potentially take a long time, and therefore, are asynchronous. For example: Making HTTP requests using fetch().

In the above code , the SetTimeout() function will be executed after a delay of 2 sec.

5. **Promises**

```
const myPromise = new Promise((resolve, reject) => {
    let success = true; // change this to false to see the rejection case
    if(success) {
        resolve('Promise was successful');
    } else {
        reject('Promise failed');
    }
});

myPromise.then(message => {
    console.log(message); // Output: "Promise was successful"
}).catch(error => {
    console.log(error); // If rejected: "Promise failed"
});
```

The **Promise** object represents the eventual completion (or failure) of an asynchronous operation and its resulting value.

The Promise object supports two properties: **state** and **result**.

While a Promise object is "pending" (working), the result is undefined.

When a Promise object is "fulfilled", the result is passed on to then() as an argument.

When a Promise object is "rejected", the result is an error object passed on to catch().

6. **Async/Await**

```
async function asyncFunc() {
    let promise = new Promise((resolve, reject) => {
        setTimeout(() => resolve('Promise resolved'), 2000);
    });

    let result = await promise;
    console.log(result); // Output: "Promise resolved"
}

asyncFunc();
```

The async/await syntax is a special syntax created to help you work with promise objects. It makes your code cleaner and clearer. When handling a Promise , you need to chain the call to the function or variable that returns a Promise using then/catch methods.

**Lab Tasks: Perform the following:**

1. Explore hoisting for let , const and var .
2. Write Arrow function that displays your name.
3. Convert a simple add function that takes two numbers and returns their sum to arrow function.

4. Implement another function processData that receives data and logs it. Pass this as a callback to getData.
5. Write a function fetchUserData that takes a callback and simulates fetching user data after 10 seconds.
6. Rewrite the getData function from above task to return a promise instead of using a callback. Chain then() and catch() to it.
7. Convert the promise chain from Lab Task 6 into an async function using await .Add Error handling using try catch.

## Experiment No. 7 Programming Constructs in PHP

**Objectives:** To familiarize students with the usage of the programming constructs in PHP

**Tools:** XAMPP Server, Dreamweaver, Browser (Internet Explorer, Google Chrome or Firefox)

**Procedure:** In order to practice the above programming constructs in PHP we will write a program to generate the following output

Figure 7.1: Expected Output

**PHP** stands for Hypertext Preprocessor. PHP is a server scripting language, and is a powerful tool for making dynamic and interactive Web pages and scripts of PHP are executed on server and result is returned to the browser as plain text. Extension for PHP file is **.php** and can create, open, delete, read, write and close files on server. PHP can collect form data and can send and receive cookies. Also, PHP can add, delete, modify data in database and can restrict users to access some pages on website. Data encryption and decryption can also be done using PHP. With PHP we are not just limited to output HTML. We can output images, PDF files, and even Flash movies and can also output any text, such as XHTML and XML.

**Step 1:** Open Dreamweaver, create New PHP page and save it as **index.php** in htdocs folder and write the following code

```
<?php

    $levels = 10;
    $output = array();
    for ($i = 1; $i <= $levels; $i = $i + 2) {
      $whitespace = ($levels - $i / 2);
      $output[] = str_pad('', $whitespace, '-') . str_pad('', $i, '*');
    }
    echo implode("\n", $output);
    echo "\n";
    echo implode("\n", array_reverse($output));
    echo "\n";
```

```
// Top of star
for ($i = 1; $i <= $levels; $i = $i + 2) {
$whitespace = ($levels - $i / 2);
echo str_pad('', $whitespace, ' ') . str_pad('', $i, '*') . "\n";
}

// Odd or even number of levels?
if ($levels % 2 == 0) {
$downlevels = $levels - 1;
} else {
$downlevels = $levels;
}

// Bottom of star
for ($i = $downlevels; $i >= 0; $i = $i - 2) {
$whitespace = ($levels - $i / 2);
echo str_pad('', $whitespace, ' ') . str_pad('', $i, '*') . "\n";
}
```

Step 2: Access the above program through XAMPP server in browser, following is the output


Figure 23: Pattern Output

**Lab Task(s):** Write a PHP program to draw the following pattern.

1. Implement the concept of integer, float, string and constant variable. Concatenate two strings and display it using echo statement.
2. Initialize a string variable as "***I am string to be tested"*** and apply string functions to reverse it, count number of words and find what is the position of word ***"be".***
3. Implement the concept of function in PHP which can return summation of two integer variables received as parameters.
4. Implement the following half diamond pattern.

```
1
12
123
1234
12345
123456
12345
1234
123
12
1
```

5. Initialize three variables of your own choice and display its value in table as shown below.

| | |
|---|---|
| First Variable value | 1111 |
| Second Variable value | 2222 |
| Third Variable value | 3333 |

6. Pretend there is a sorted list such as 1, 1, 2, 2, 3, 4, 5, 5. Write code that could remove all the duplicated from the list and make it as 1, 2, 3, 4, 5.
7. Find all the prime numbers that are less than 100 and compute its summation using PHP.
8. Implement the concept of Associative Arrays.
9. Implement a function in PHP which can take five-digit integer as argument and return its summation. For example, the summation of 12345 is 15.
10. Create an array with the following values: Pakistan, England, India, America, Dubai, Saudi Arabia, Mexico, Turkey, Holland, Karachi, Peshawar, Lahore. Print these values to the browser separated by commas, using a loop to iterate over the array. Sort the array, then print the values to the browser in an unordered list, again using a loop. Add the following cities to the array: Quetta, Faisalabad and Multan. Sort the array again, and print it once more to the browser in an unordered list.

## Experiment No. 8 Connectivity with Database Server

**Objectives:** To familiarize students with database connectivity through web server and to upload different format of files to the server (localhost)

**Tools:** XAMPP Server, Dreamweaver, Browser (Internet Explorer, Google Chrome or Firefox)

**Procedure:** To connect a web page to MySQL database Server and insert data to it from web page, the following steps must be followed

In order to insert, delete, update or manipulate data in database, we need to establish connection with database server. Database connectivity and retrieval of data can be done in a number of steps such as

- Make the connection
- Select database
- Perform queries on the table
- Print out the data

We need server name, MySQL username, password in order to establish connection with server. Create a file as connection.php file, open and close the php code with tags and write MySQL username, password and server name. For queries, that is to add data in database, we select corresponding table in database and with the help of loop retrieve data in order to display it in a web browser. For insertion, deletion and edit data, we use *insert, delete and update* queries respectively.

**File uploading -** A PHP script can be used with a HTML form to allow users to upload files to the server. Initially files are uploaded into a temporary directory and then relocated to a target destination by a PHP script. Information in the phpinfo.php page describes the temporary directory that is used for file uploads as upload_tmp_dir and the maximum permitted size of files that can be uploaded is stated as upload_max_filesize. These parameters are set into PHP configuration file php.ini.

The process of uploading a file follows these steps

- The user opens the page containing a HTML form featuring a text files, a browse button and a submit button.
- The user clicks the browse button and selects a file to upload from the local PC.
- The full path to the selected file appears in the text filed then the user clicks the submit button.
- The selected file is sent to the temporary directory on the server.
- The PHP script that was specified as the form handler in the form's action attribute checks that the file has arrived and then copies the file into an intended directory.
- The PHP script confirms the success to the user.

As usual when writing files it is necessary for both temporary and final locations to have permissions set that enable file writing. If either is set to be read-only then process will fail. An uploaded file could be a text file or image file or any document.

**Step 1:** Open the Dreamweaver and write the following PHP code

```
<?php

    if(isset($_REQUEST['ck']) and $_REQUEST['ck'] == 1)
    {
        echo $qu = "insert into user set
                                    user_name = '$_REQUEST[name]',
                                    user_gender = '$_REQUEST[gender]',
                                    user_address = '$_REQUEST[add]',
                                    user_email = '$_REQUEST[email]',
                                    login_id = '$_REQUEST[login]',
                                    password = '$_REQUEST[pass]'  ";
    if(execute_query($qu))
    {
         echo "<h1>Record has been Successfully inserted into the database</h1>";
    }
    }
    function execute_query($query){
    $connection = mysql_connect("localhost","root","");
    mysql_select_db("lab",$connection);
    $result_set = mysql_query($query);
    mysql_close($connection);
    return $result_set;
?>
    <html>
    <script language="javascript">
    function validation()
    {
      if(document.getElementById("name").value == '' ”)
      {
      alert("Please Enter Name");
      return false;
    }
    else if(document.getElementById("email").value == '' “)
    {
      alert("Please Enter Email");
      return false;
    }
    else if(document.getElementById("add").value == '' “)
    {
      alert("Please Enter Adrees");
```

```
    return false;
  }
  else if(document.getElementById("login").value == '' “)
  {
    alert("Please Enter Login ID");
    return false;
  }
  else if(document.getElementById("pass").value == '' “)
  {
    alert("Please Enter Password");
    return false;
  }
  else if(!document.getElementById("chk").checked)
  {
    alert("Please Agree with Policy");
    return false;
  }
  }
  </script>

  <head><title>Web Engineering Lab 06</title></head>
  <body>
    <h2>Sign Up Form</h2>
    <form action="" method="post" onSubmit="return validation();" >
    <input type="hidden" name="ck" value="1">
    <table>
      <tr>
         <td width="10%">Name:</td>
         <td width="90%"><input type="text" name="name" id="name" ></td>
      </tr>

      <tr>
          <td>Gender:</td>
          <td><input type="radio" name="gender" value="male" checked >Male <br>
          <input type="radio" name="gender" value="female" >Female</td>
      </tr>

      <tr>
           <td>Email:</td>
           <td><input type="text" name="email" id="email" ></td>
      </tr>
```

```
        <tr>
            <td>Address:</td>
            <td><textarea name="add" id="add" ></textarea></td>
        </tr>

        <tr>
            <td>Login ID:</td>
            <td><input type="text" name="login" id="login" ></td>
        </tr>

        <tr>
            <td>Password:</td>
            <td><input type="password" name="pass" id="pass" ></td>
        </tr>

        <tr>
            <td>Agreed with<br> Policy:</td>
            <td><input type="checkbox" name="agree" id="chk" ></td>
        </tr>
        <tr>
        <td></td>
        <td ><input type="submit" value="Submit">   
        <input type="reset" value="Reset"></td>
        </tr>
  </table>
  </form>
</body>
</html>
```

**Step 2:** The program will look in the browser as follow, please fill the form and click on Submit button


Figure 8.1: Form for data insertion

**Step 3:** The Record is successfully inserted into the database and the success message is shown in the browser as follow

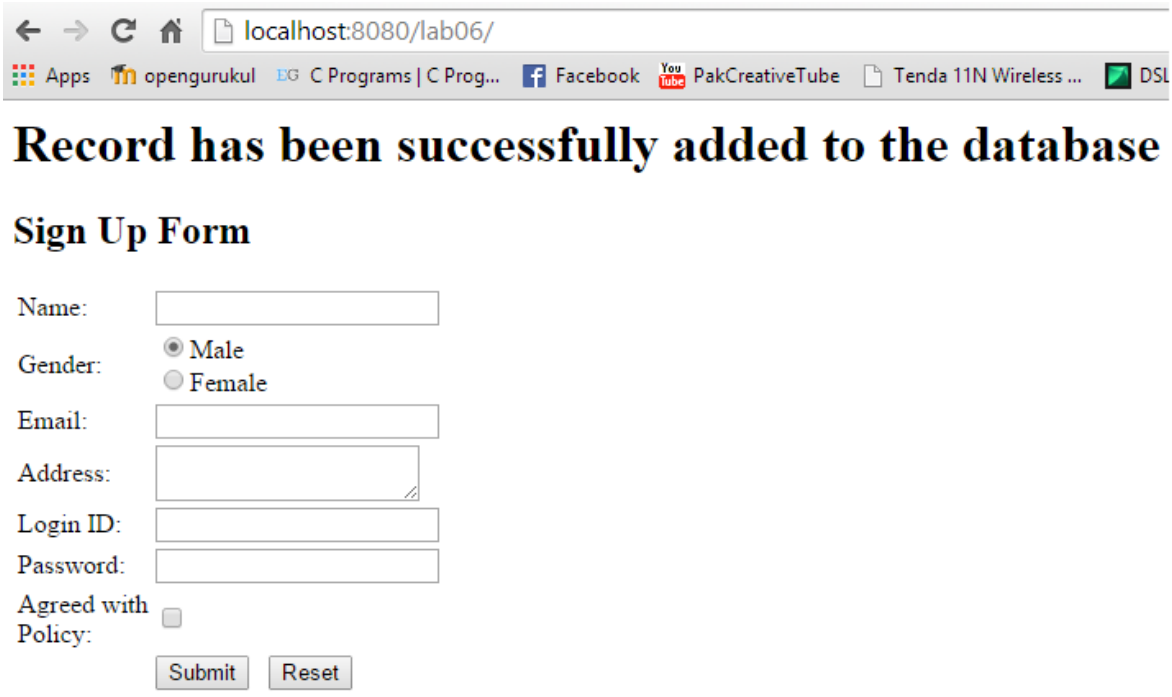



Figure 8.2: Shot after successful insertion

**Step 4:** To ensure the presence of this record in database, open MySQL Server front end, the highlighted is newly inserted record in the following snapshot

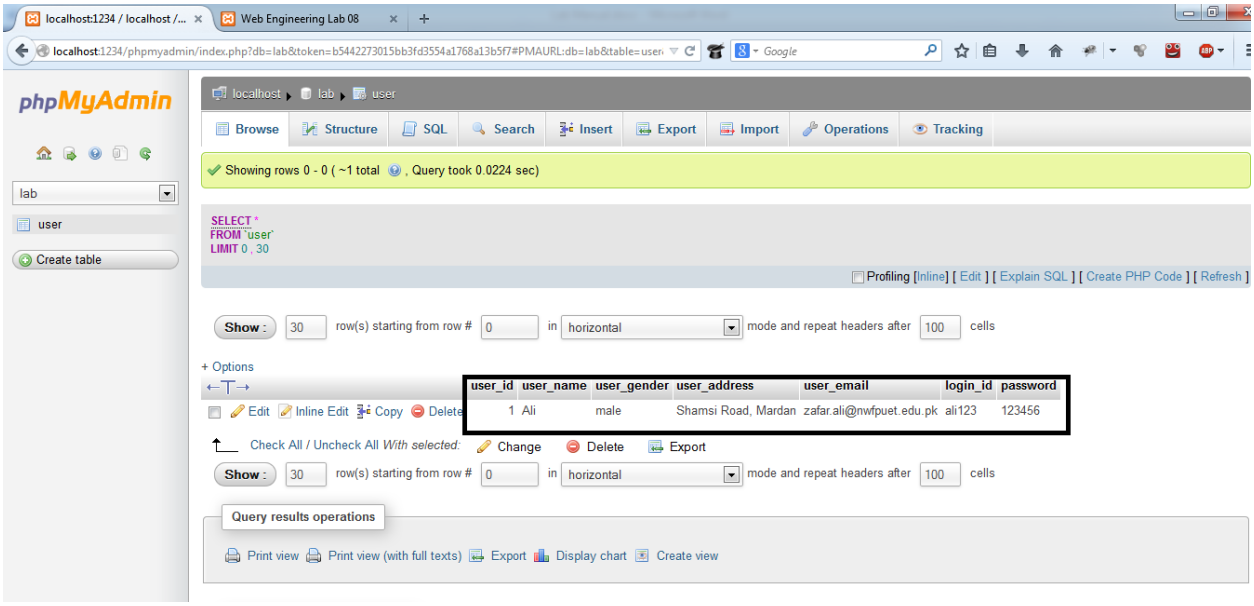



**Figure 8.3: New inserted record, MySQL Server view**

**Step 5:** Create another PHP file in uetmardanlabs folder as "fileupload.php" and write the following code in it.

```
<?php

		$db_host = "localhost";
		$db_username = "root";
		$db_password = "";
		$db_database = "uetmardan";

		$connection = mysql_connect($db_host,$db_username,$db_password);
		if(!$connection) die("Unable to connect to server".mysql_error());
		$db_server = mysql_select_db($db_database,$connection);
		if(!$db_server) die("Unable to connect to database".mysql_error());

			// Upload File
			if(isset($_POST['hidden']))
			{
			if ((($_FILES["file"]["type"] == "image/gif")
					|| ($_FILES["file"]["type"] == "image/jpeg")
					|| ($_FILES["file"]["type"] == "image/pjpeg")
					|| ($_FILES["file"]["type"] == "image/png"))
					&& ($_FILES["file"]["size"] < 2000000))   //less than 20kb
			{
			    if ($_FILES["file"]["error"] > 0)
```

```
                                {
                                  echo "Return Code: " . $_FILES["file"]["error"] . "<br />";
                                }
                        else
                         {
                                echo "Upload: " . $_FILES["file"]["name"] . "<br />";
                                echo "Type: " . $_FILES["file"]["type"] . "<br />";
                                echo "Size: " . ($_FILES["file"]["size"] / 1024) . " Kb<br />";
                                echo "Temp file: " . $_FILES["file"]["tmp_name"] . "<br />";

                                if (file_exists("uploads/" . $_FILES["file"]["name"]))
                                        {
                                                echo $_FILES["file"]["name"] . " already exists. ";
                                        }
                        else
                        {
                                move_uploaded_file($_FILES["file"]["tmp_name"],
                                                "uploads/" . $_FILES["file"]["name"]);
                                echo "Stored in: " . "uploads/" . $_FILES["file"]["name"];
                        }
                        }
                                }
                        else
                                {
                                        echo "Invalid file";
                                }
                        }
?>

<html>
<body>

    <form name="addform" method="post" action="fileupload.php" enctype="multipart/form-data">

            Name        <input type="text" name="addcategory"><br>
                        <input type="file" name="image"><br>
                        <input type="hidden" name="hidden" value="hidden"><br>
                        <input type="submit" name="enter" value="Upload File ">
    </form>

</body>
</html>
```

**Step 6:** Provide “localhost:8080/uetmardanlabs/fileupload.php” in url to view the following figures output.

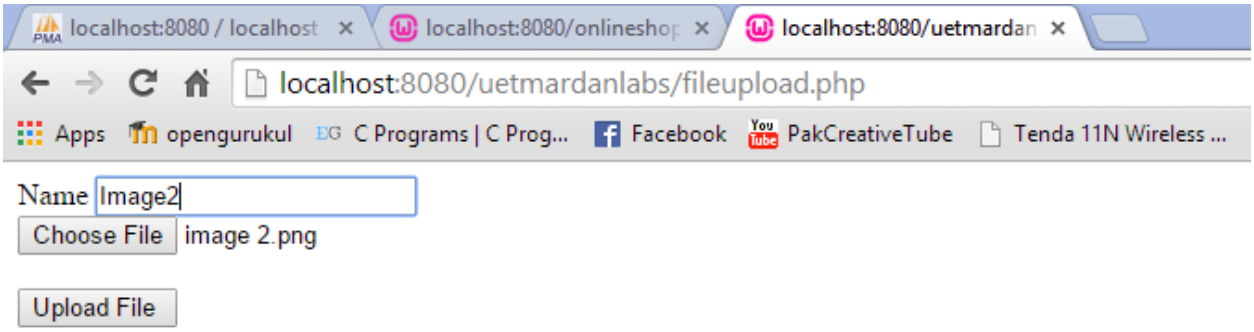



Figure 8.4: File Upload

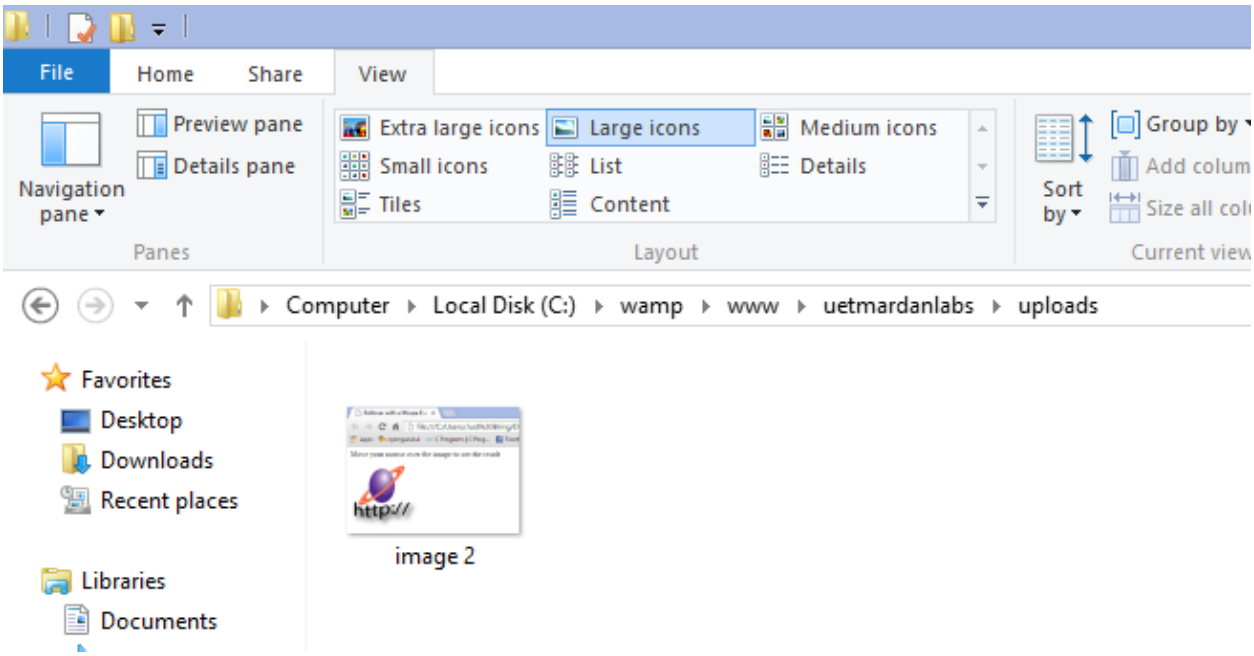



Figure 8.5: File Uploaded

**Lab Task(s):** Perform the following:

1. Create sign up form as shown in figure 01 and establish connection with server and database.
2. Insert form data in database as shown in figure 28.
3. Update figure 01 form data into database
4. Retrieve and display form data from database
5. Delete form data in figure 01 from database


Figure 8.6: Sign Up Form

6. Write PHP and HTML code to upload file or image to server (localhost).

## Experiment No. 9 Session maintenance in HTTP

**Objectives:** To familiarize students with state maintenance in PHP that is how to create and maintain user login or sign in form in dynamic websites

**Tools:** WAMP Server, Dreamweaver, Browser (Internet Explorer, Google Chrome or Firefox)

**Procedure:** Creating PHP page which will perform the following tasks

1. To create sign-in form in order to check data similarity provided by user in a form against data already store in database in order to access main page in a website or her account in a website
2. To demonstrate sessions using shopping cart functinalities

The data as username and password for login form is already inserted in database table as admin and admin respectively.

Often website have registration form through which users register herself and then sign in again in order to use web pages. Through registration form, data of user is inserted into database.

**Sessions -** A normal HTML website will not pass data from one page to another. In other words, all information is forgotten when a new page is loaded. This makes it quite a problem for tasks like a shopping cart, which requires data (the user's selected product) to be remembered from one page to the next. A PHP session solves this problem by allowing you to store user information on the server for later use (i.e. username, shopping cart items, etc.). However, this session information is temporary and is usually deleted very quickly after the user has left the website that uses sessions. It is important to ponder if the sessions' temporary storage is applicable to your website. If you require a more permanent storage you will need to find another solution, like a MySQL database. Sessions work by creating a unique identification (UID) number for each visitor and storing variables based on this ID. This helps to prevent two users' data from getting confused with one another when visiting the same webpage.

Before you can begin storing user information in your PHP session, you must first start the session. When you start a session, it must be at the very beginning of your code, before any HTML or text is sent. Below is a simple script that you should place at the beginning of your PHP code to start up a PHP session.

```
<?php session_start(); ?>
```

The above piece of code will register the user's session with the server, allow you to start saving user information and assign a UID (unique identification number) for that user's session. When you want to store user data in a session use the **$_SESSION associative array**. This is where we both store and retrieve session data. The sessions can be clean by using ***unset($_session[$var])*** and can be delete permanently by using ***session_destroy()*** function.

**Log-in form** is the procedure to get access to an operating system or application, usually in a remote computer. Almost always a log-in requires that the user have an ID and a password. The user ID can be of many characters and the password must contain at least one digit. The user ID can be freely known and is visible when entered at a keyboard or other input device. The password must be kept secret and is not

displayed as it is entered. The provided data by user at time of registration is checked against login-form in order to use web pages.

**Step 1:** Open Dreamweaver and write the following PHP code and save it as index.php in root folder "uetmardanlabs".

```
<?php

                $db_host = "localhost";
                $db_username = "root";
                $db_password = "";
                $db_database = "uetmardan";

                $connection = mysql_connect($db_host,$db_username,$db_password);
                if(!$connection) die("Unable to connect to server".mysql_error());
                $db_server = mysql_select_db($db_database,$connection);
                if(!$db_server) die("Unable to connect to database".mysql_error());

                // Sign in code

                $user_name = $_POST['usertxtbox'];
                $user_pass = $_POST['passtxtbox'];

                if(isset($_POST['signinbutton']))
                {
                        $sql = "SELECT * FROM login WHERE username = '".$user_name."'
                                and password =    '".$user_pass."'";
                         $result = mysql_query($sql);

                           while($rows = mysql_fetch_array($result))
                           {
                                    if($rows['username'] == $user_name and
                                     $rows['password'] == $user_pass)
                                       {
                                                echo "Welcome to the screen";
                                                echo "<br><br>";
                                        }
                                      else
                                       {
                                                echo "sorry wrong information provided";
                                                echo "<br><br>";
                                       }
                                }
                        }

  ?>
```

```
<form name="signin" method="post" action="index.php">

Username      <input type="text" name="usertxtbox"><br>
Password      <input type="password" name="passtxtbox"><br>
              <input type="submit" name="signinbutton" value="Sign In">
</form>
```

**Step 2:** Now open web browser and provide "localhost:8080/uetmardanlabs" in url. Provide username as "admin" and password as "admin" to see the required output as shown in figures below.

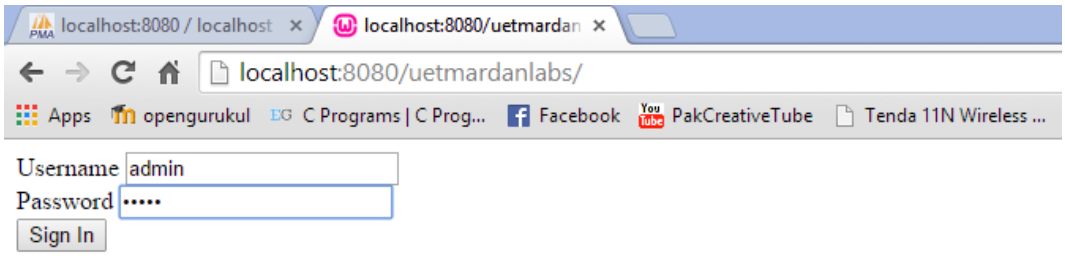



Figure 9.1: PHP Login File

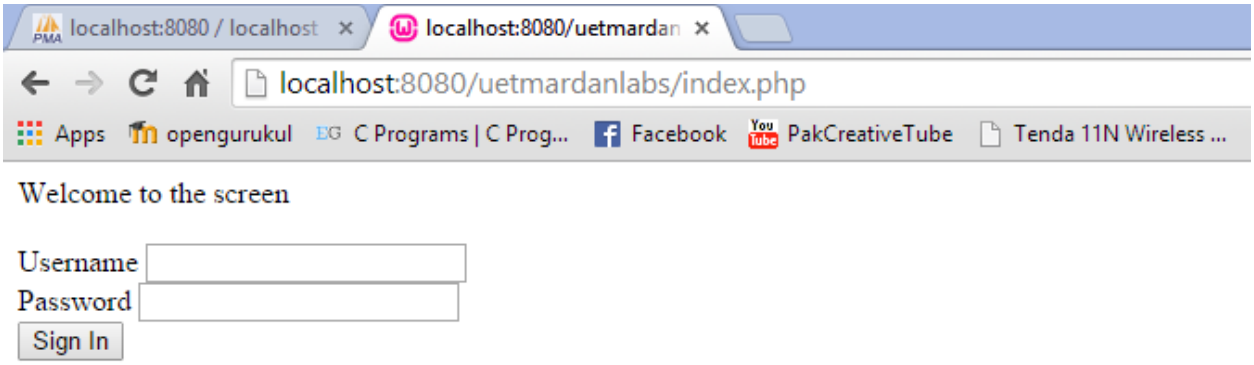



Figure 9.2: PHP Login File (User Signed In)


Figure 9.3: User Database Data

**Step 3:** Create another new file in Dreamweaver and write the following code and save it as checklogin.php

```
<?php session_start();

        include("connection.php");

 // if session exists
 if(isset($_SESSION['admin']) && $_SESSION['admin'] != '')
        {
                echo "some one is already logged in please first logout";
                echo "<a href='logout.php'>Logout</a>";
        }
// Database and user value checking
 else
        {
                $query = "SELECT * FROM logintable WHERE
                        username = '".$_POST['admintextbox']."' AND
                        password = '".$_POST[ 'adminpassword']."' AND user_type = '1'";

                 $rows = mysql_query($query);
                $data = mysql_fetch_array($rows);

                if($_POST['admintextbox'] == $data['username'] and
                  $_POST['adminpassword'] == $data['password'])
```

```
                        {
                                // Assign session to user
                            $_SESSION['admin'] = $_POST['admintextbox'];

                                echo "Welcome ".$_POST['admintextbox'];

                                echo “<input type = button onClick =
                                    ‘window.javascript.location =
                                    “showcategory.php”’>Categories</button>”;
                                    echo "<a href='logout.php'>Log Out</a>";
                        }
                    else
                                {
                                    echo "you are not admin";
                                }
            }
?>
```

**Step 4:** Create another new file in Dreamweaver and write the following code in it and save it as showcategory.php

```
<?php session_start();
            include("connection.php");

            // find out how many rows are in the table
            $sql = "SELECT COUNT(*) FROM category";
            $result = mysql_query($sql);
            $r = mysql_fetch_row($result);
            $numrows = $r[0];

            // number of rows to show per page
            $rowsperpage = 2;
            // find out total pages
            $totalpages = ceil($numrows / $rowsperpage);

            // get the current page or set a default
            if (isset($_GET['currentpage']) && is_numeric($_GET['currentpage'])) {
              // cast var as int
              $currentpage = (int) $_GET['currentpage'];
            } else {
              // default page num
              $currentpage = 1;
            } // end if

            // if current page is greater than total pages...
            if ($currentpage > $totalpages) {
```

```
                  // set current page to last page
                  $currentpage = $totalpages;
                } // end if
                // if current page is less than first page...
                if ($currentpage < 1) {
                  // set current page to first page
                  $currentpage = 1;
                } // end if

                // the offset of the list, based on current page
                $offset = ($currentpage - 1) * $rowsperpage;

                // get the info from the db

                $query = mysql_query("SELECT * FROM category LIMIT $offset, $rowsperpage");
                echo "<table border=1>";
                 echo "<th>ID</th><th>Category Name</th><th>Number of Products </th>
                 <th>Thumbnail</th><th>Delete</th><th>Edit</th>";

                while($data  = mysql_fetch_array($query))
                {
                   echo "<tr><td>";
                   echo $data['id'];
                          echo "</td><td>";?>

      <a href = "showproducts.php?category=<?php echo $data['name']?>">
      <?php echo $data['name']?></a>

<?php
                echo "</td><td>";
                $query1     = "SELECT * FROM product WHERE category='".$data['name']."'";
                $numofprod  = mysql_num_rows(mysql_query($query1));
                echo $numofprod."</td><td>";?>

               <img src="categorythumbs/<?php echo $data['thumbnail']; ?>" />
<?php
                echo "</td><td>";
?>

       <a href = "deletecategory.php?id=<?php echo $data['id'] ?>&category=<?php echo
      $data['name']?>">Delete</a>

 <?php

        echo "</td><td>";
 ?>
```

```
    <a href = "editcategory.php?id=<?php echo $data['id'] ?>&category=<?php echo $data['name']
    ?>">Edit</a>

<?php

        echo "</td></tr>";

?>

<?php

        }
                echo "</table>";

                        /*   build the pagination links  */
                // range of num links to show
                $range = 3;

                // if not on page 1, don't show back links
                if ($currentpage > 1)
                 {

                  // show << link to go back to page 1
                  echo " <a href='{$_SERVER['PHP_SELF']}?currentpage=1'><<<</a> ";
                  // get previous page num
                  $prevpage = $currentpage - 1;
                  // show < link to go back to 1 page
                  echo " <a href='{$_SERVER['PHP_SELF']}?currentpage=$prevpage'><</a> ";
                  } // end if

                // loop to show links to range of pages around current page
                for ($x = ($currentpage - $range); $x < (($currentpage + $range) + 1); $x++)
                 {
                  // if it's a valid page number...
                  if (($x > 0) && ($x <= $totalpages))
                   {
                      // if we're on current page...
                      if ($x == $currentpage)
                      {
                         // 'highlight' it but don't make a link
                            echo " [<b>$x</b>] ";
                             // if not current page...
                      }
                     else
                      {
                            // make it a link
                            echo " <a href='{$_SERVER['PHP_SELF']}?currentpage=$x'>$x</a> ";
```

```
                    } // end else
                  } // end if
                } // end for

              // if not on last page, show forward and last page links

              if ($currentpage != $totalpages)
               {
                 // get next page
                 $nextpage = $currentpage + 1;

                // echo forward link for next page
                 echo " <a href='{$_SERVER['PHP_SELF']}?currentpage=$nextpage'>>></a> ";

               // echo forward link for lastpage
               echo " <a href='{$_SERVER['PHP_SELF']}?currentpage=$totalpages'>>>></a> ";
               } // end if

              /*  end build pagination links */

?>

<html>
<body>
              <form method="post" action="addcategory.html"><br>
              <input type="submit" name="enter" value="Add Category" />
              </form> <br /><br />

              <a href="category.php">Home</a>
</body>
</html>
```

**Step 5:** Create a new file in Dreamweaver, write the following code in it and save it as logout.php

```
<?php session_start();

          include("connection.php");
           unset($_SESSION['admin']);
           unset($_SESSION['user']);
           echo "you are logged out<br><br>";
           include("index.php");

?>
```

**Step 6:** Open web browser, provide as "localhost/onlineshop/administrator/" in url to see the following required output as shown in figures below.

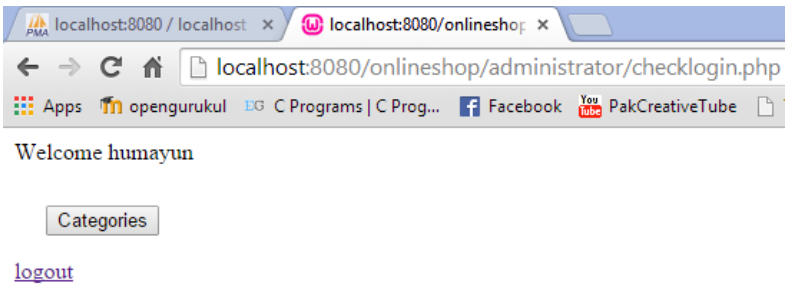



Figure 9.4: Welcome User after sign in

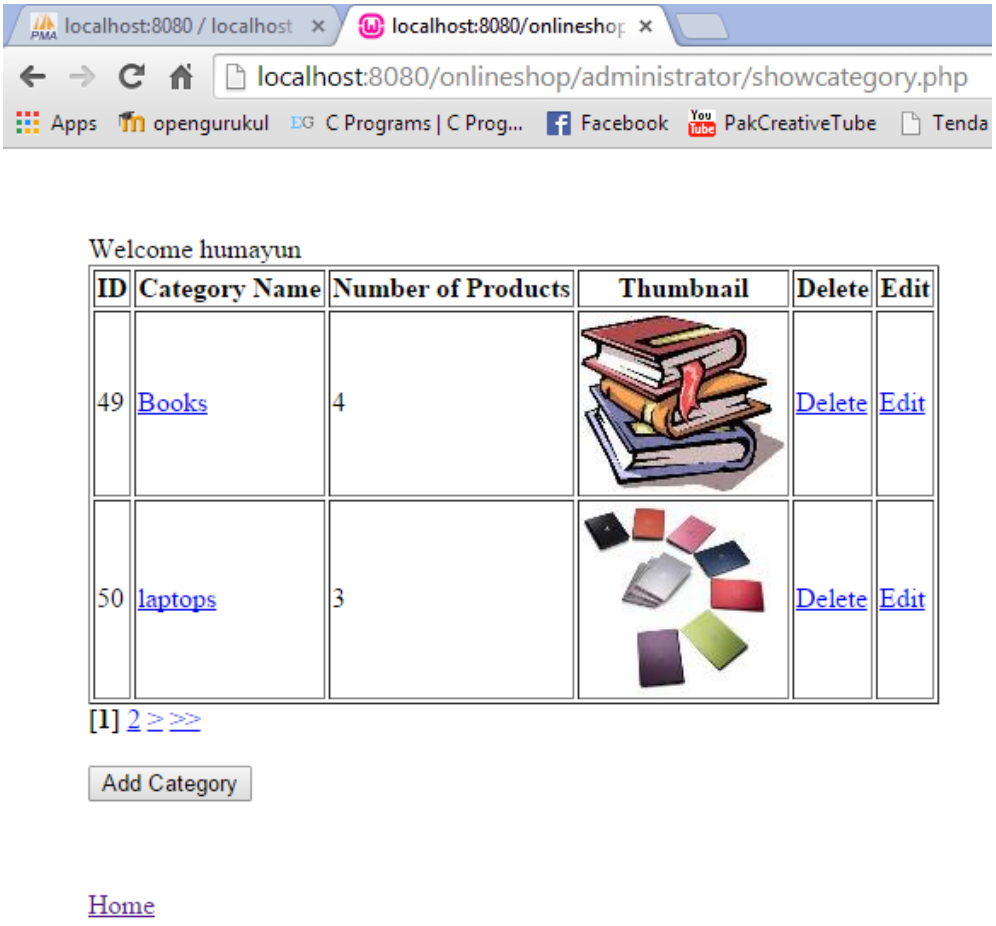



Figure 9.5: Record of items in Shopping cart

**Lab Tasks(s):** Perform the following tasks

1. Implement login system using sessions
2. Create a category page as mentioned in corresponding lab manual lab, and add three categories of your own choice
3. Add items to category of your own choice
4. Display all the categories items in tabular form

## Experiment No. 10 Configuration of React JS Library and exploring react application folder structure

**Objectives:** To familiarize students with React JS Library and its different components

**Tools:** VS Code, Browser (Internet Explorer, Google Chrome or Firefox)

**Procedure:** To demonstrate as how to configure React JS library and exploring its application folder structure

**React.js** is a JavaScript library developed by Facebook for building user interfaces, particularly for single-page applications (SPAs) and web applications that require a dynamic and responsive user interface. It's known for its component-based architecture, which allows developers to build encapsulated UI components and manage their state efficiently.

**Features of React JS**

- **Component-Based Architecture:** React.js follows a component-based approach where the UI is divided into reusable components. Each component manages its own state and can be composed together to build complex UIs.
- **Virtual DOM:** React.js uses a virtual DOM (Document Object Model) to improve performance. Instead of directly manipulating the DOM, React creates a lightweight virtual representation of it in memory and updates only the necessary parts when the state of components changes. This minimizes the number of DOM manipulations, resulting in faster rendering.
- **JSX (JavaScript XML):** JSX is a syntax extension for JavaScript that allows developers to write HTML-like code within JavaScript. It makes the code more readable and intuitive, and it's transpired to regular JavaScript by tools like Babel.
- **One-Way Data Binding:** React.js uses one-way data binding, which means data flows only in one direction, from parent components to child components. This makes the code predictable and easier to debug.
- **Declarative Syntax:** React promotes a declarative programming style where developers describe how the UI should look based on the current application state, rather than imperatively manipulating the DOM to achieve the desired UI changes.
- **Lifecycle Methods:** React components have lifecycle methods that allow developers to hook into different stages of a component's lifecycle, such as mounting, updating, and unmounting. This enables developers to perform tasks like fetching data, updating the UI, and cleaning up resources at the appropriate times.
- **React Hooks:** Introduced in React 16.8, hooks are functions that allow developers to use state and other React features in functional components without writing a class. Hooks provide a more concise and readable way to manage state and side effects in functional components.

**Applications of React.js:**

- **Single-Page Applications (SPAs):** React.js is commonly used for building SPAs where the entire application runs in a single web page, providing a seamless user experience similar to that of a desktop application.
- **User Interfaces (UIs) for Web Applications:** React.js is suitable for building dynamic and interactive user interfaces for various web applications, including social media platforms, e-commerce websites, content management systems, and more.
- **Mobile App Development:** React Native, a framework built on top of React.js, allows developers to build cross-platform mobile applications for iOS and Android using JavaScript and React principles. React Native enables code reusability between web and mobile platforms, reducing development time and effort.
- **Progressive Web Apps (PWAs):** React.js can be used to build PWAs, which are web applications that offer a native app-like experience, including offline capabilities, push notifications, and home screen installation. React's component-based architecture and performance optimization make it well-suited for PWAs.
- **Enterprise Applications:** React.js is widely adopted by enterprises for building complex and scalable web applications due to its modular architecture, performance optimization, and robust ecosystem of libraries and tools.

There are different IDEs which can be used to implement the React application. In this lab, I will use the VS Code with few of the important extensions for ease. List of the useful extensions are:

1. Babel extension for the syntax highlights color etc. of the JavaScript
2. JavaScript ES6 code snippets, it provides the shortcuts command to the longest format code
3. VScode icons
4. Bracket highlighter (opening and closing of the code)
5. Auto rename tag (the closing tag automatically rename itself after change)

Before building the React Application, there is a need to install the Node.js and npm (Node Package Manager). Node.js is a JavaScript runtime environment that allows to run JavaScript on the server-side. Node Package Manager (npm) is a package manager for JavaScript that comes bundled with Node.js and is used to install and manage dependencies for the project.

Install Node.js by visiting the Node JS website (https://nodejs.org/en) and install the LTS(long time support) version as shown in the figure below


**Figure 10.1 Installation of Node.js Long Term Support**

To check the node and npm version, we can use the command line of the operating system or the VS Code terminal as shown below

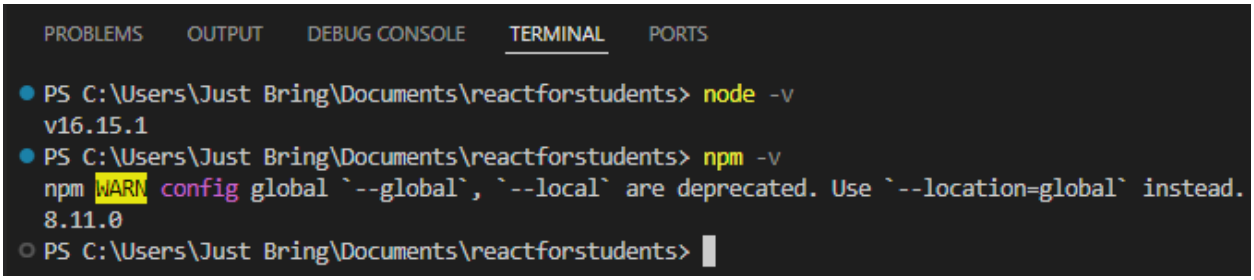



**Figure 10.2 Installation of Node.js and npm verification**

Now create empty folder in any directory, here it is This PC -> Documents. The folder will have all the react related projects. Now open the same folder in the VS Code and open the new terminal for creating new react project in the folder as shown in the figures below:

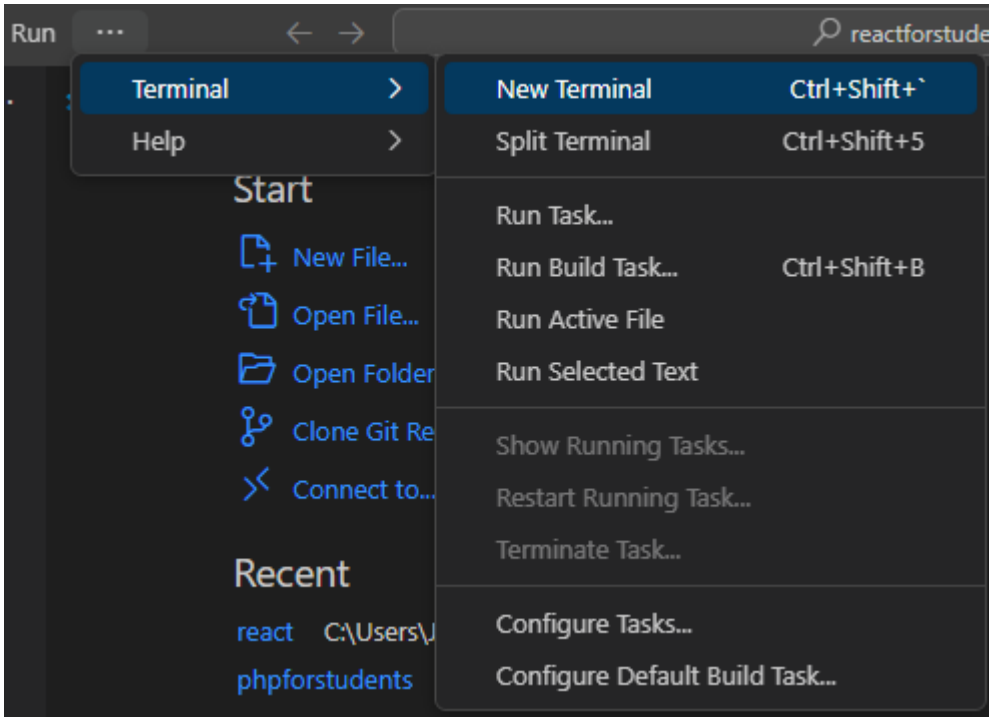



**Figure 10.3 Opening of New Terminal in VS Code to create React Application**

To create the new react app use the command as ***npx create-react-app "your app name".*** for example, ***npx create-react-app first-app*** as shown below

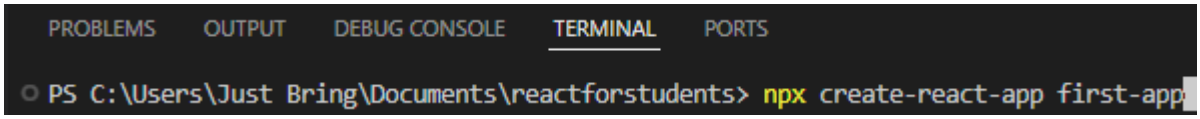



**Figure 10.4 Command to create new react application**

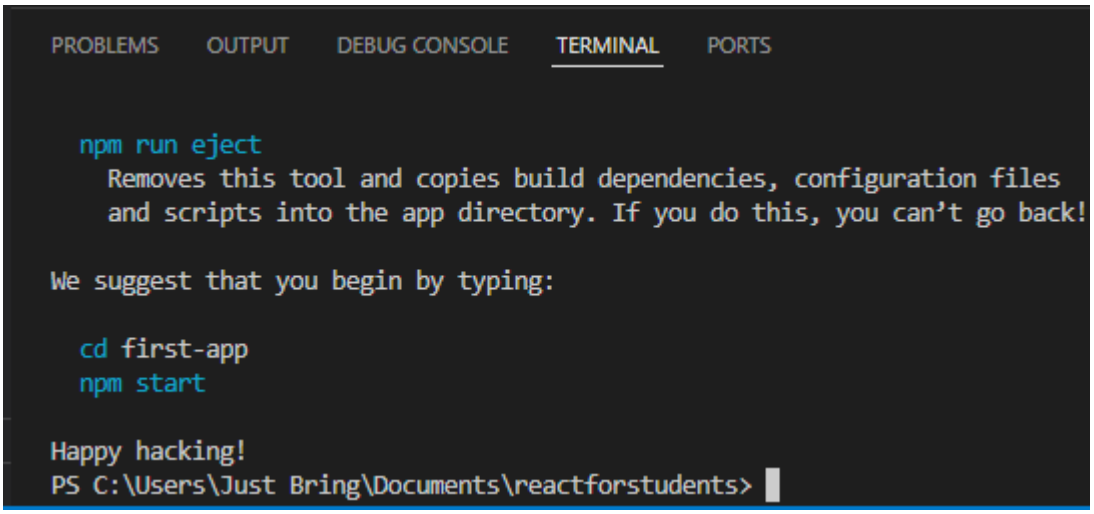



**Figure 10.5 React Application created successfully**

The project with the name first-app is created in the directory Documents -> reactforstudents -> first-app as shown below

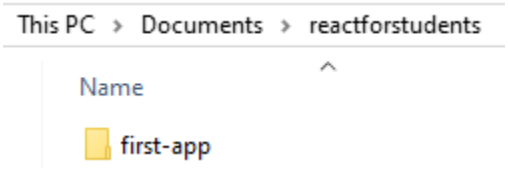



**Figure 10.6 React Application created in Documents**

To execute the project on the localhost server, redirect to the created project which is first-app in this case. For redirection, use the commands ***cd first-app*** and then ***npm start*** to execute on the localhost server as shown below

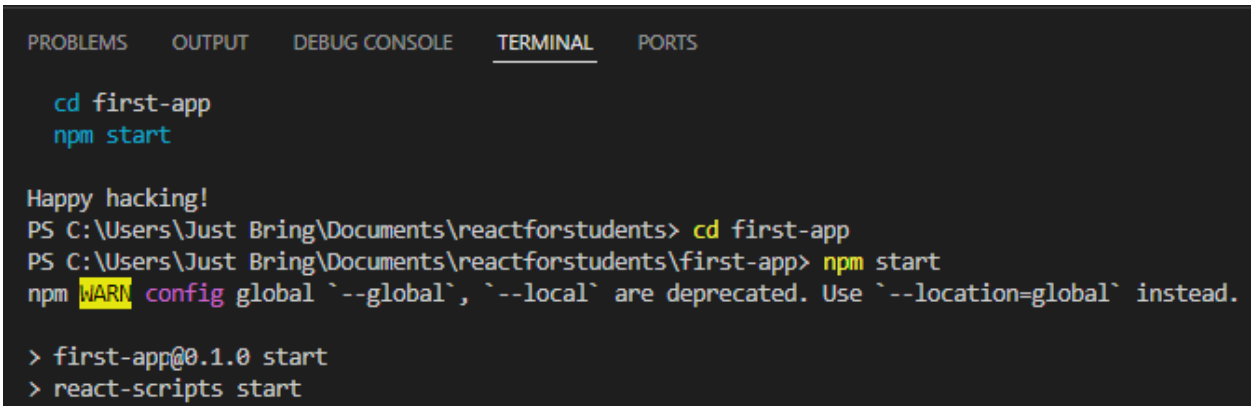



**Figure 10.7 React Application execution commands in the terminal**

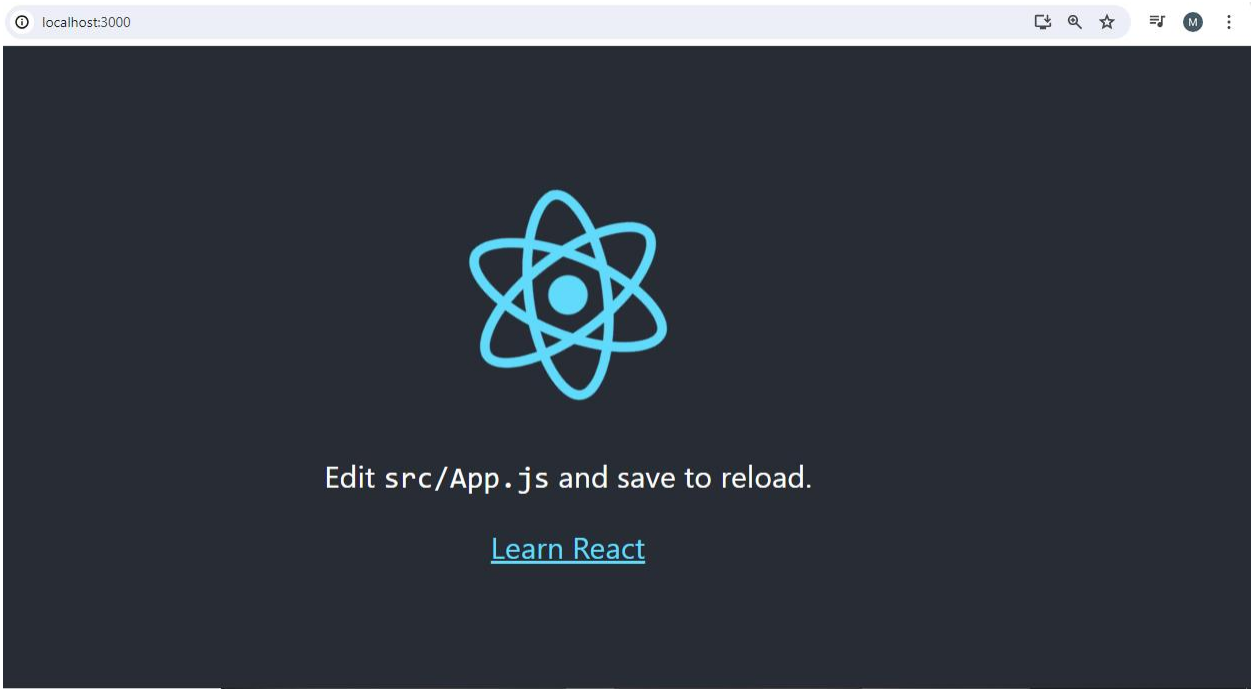



**Figure 10.8 React Application executed locally successfully**

**Workflow of the react app folder structure**

The initial folder structure of the react application is as follow:

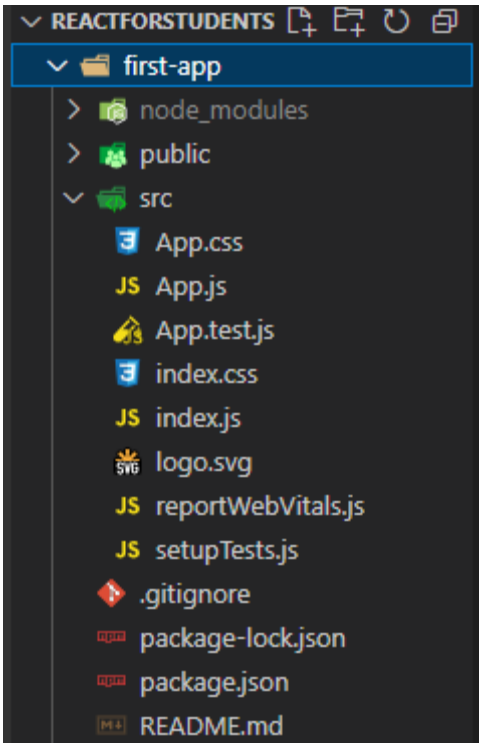


**Figure 10.9 React Application folder structure**

There are three most important files which used to fetch and render/display the data. As the react is based on the components. The file public -> index.html is used to render the data on the browser while the file src -> index.js is used to fetch the data from the component src -> App.js. The index.js served as a middle man which fetched the data from the components and sends component data to the render page which is part of the public folder having root id. The process can be shown in the following figures.

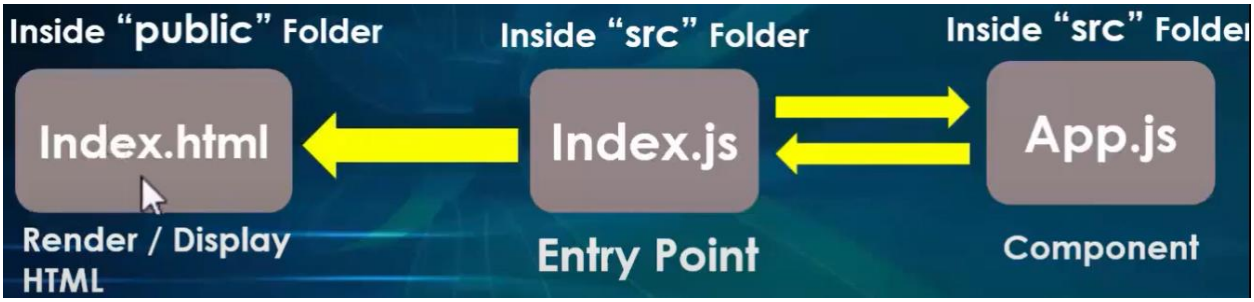



**Figure 10.10 React Application rendering between component and browser display**

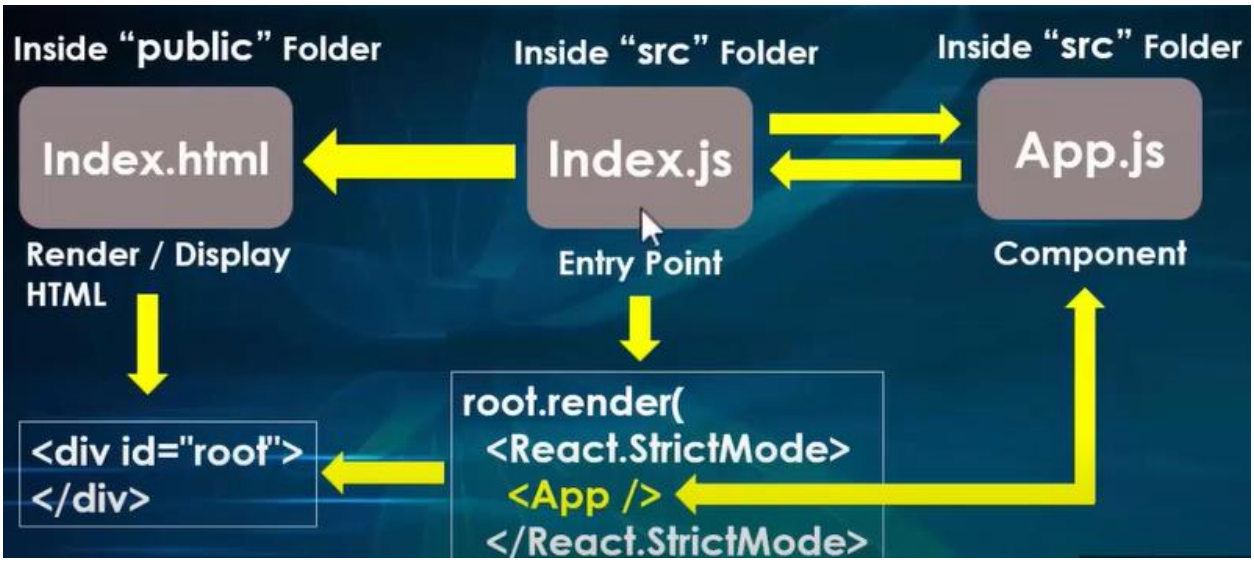



**Figure 10.11 React Application rendering between component and browser display with code**

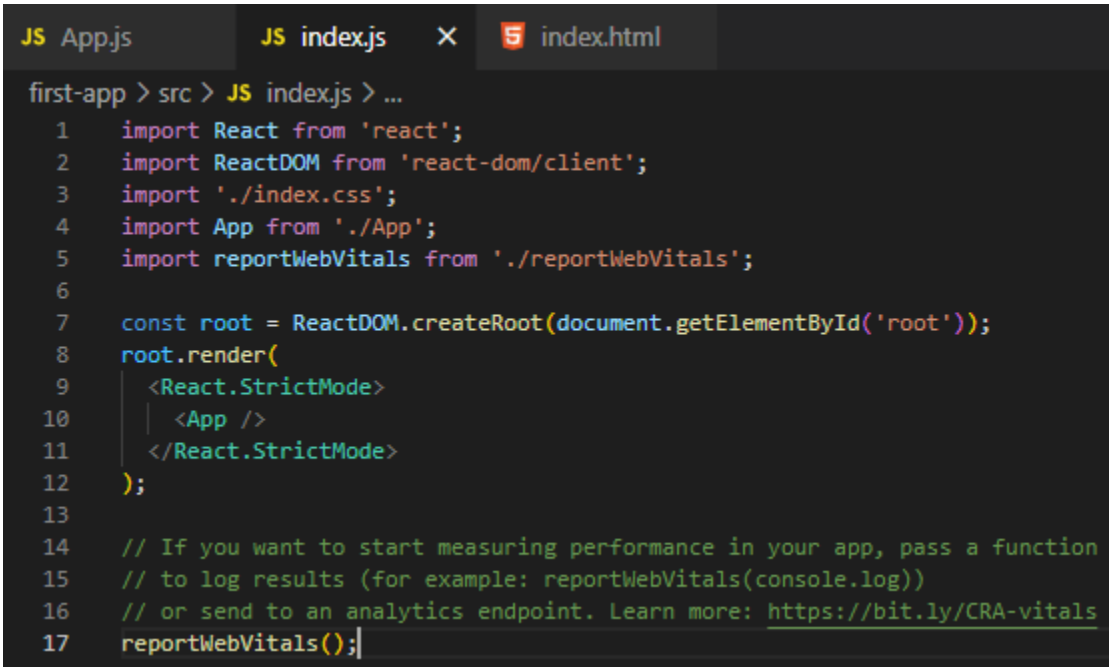



**Figure 10.12 React Application index.js code**

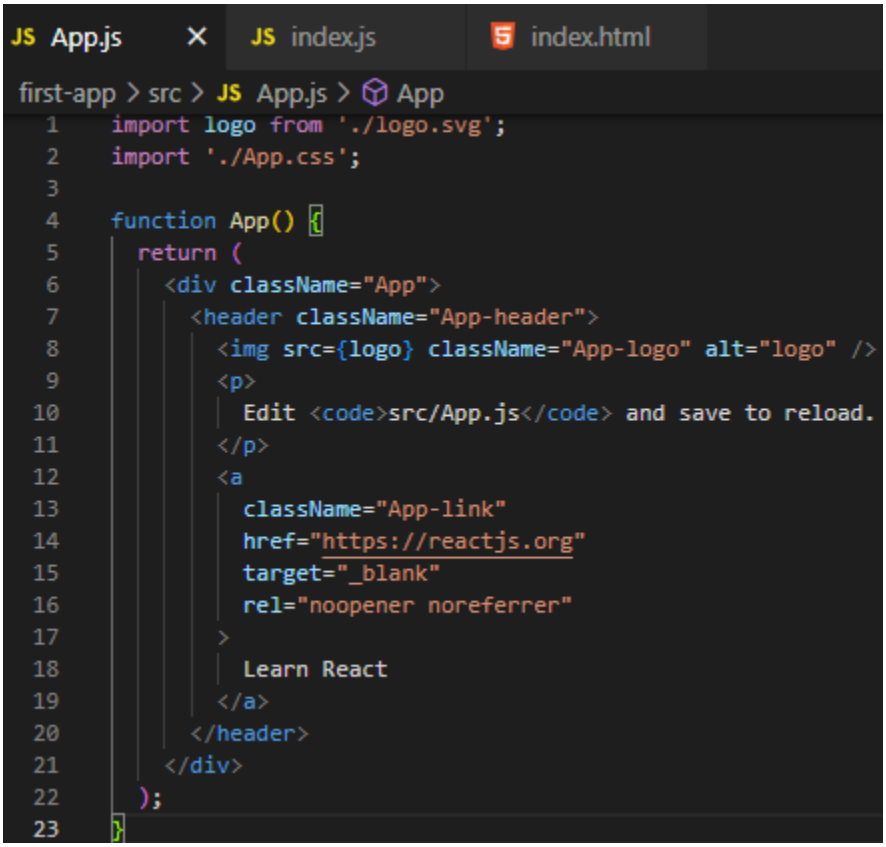



**Figure 10.12 React Application component App.js code**

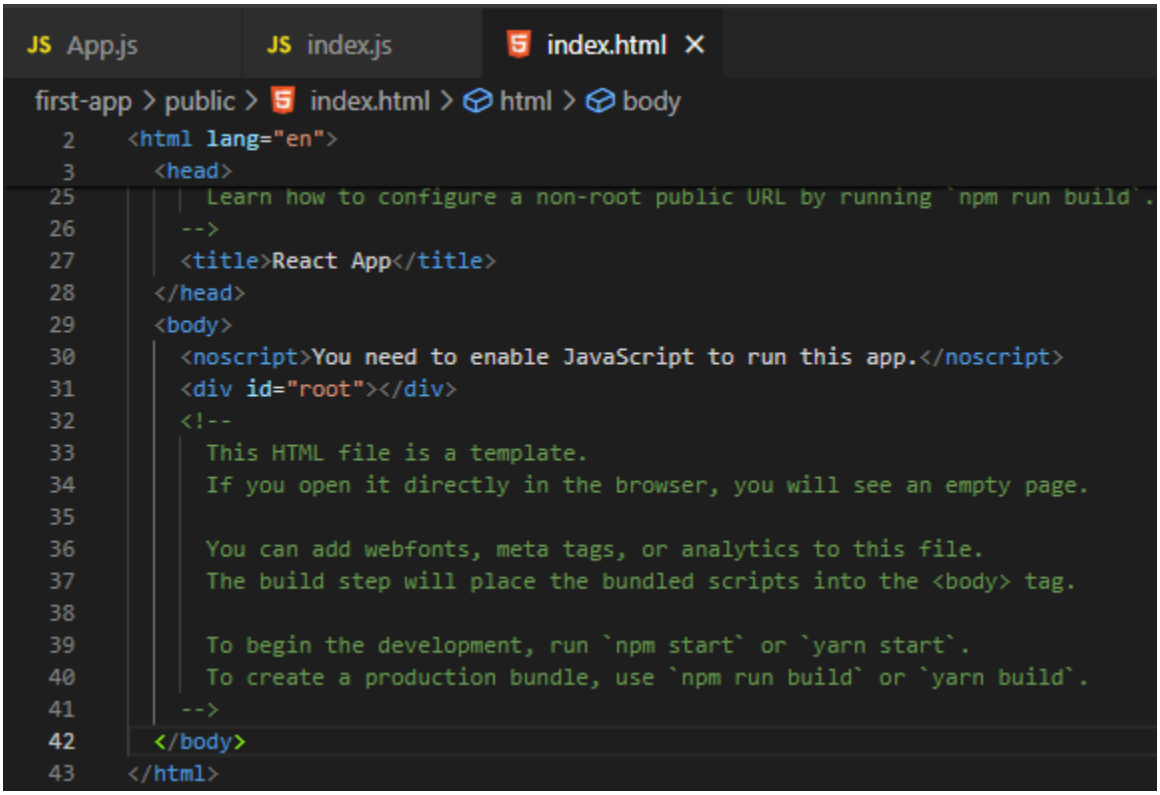



**Figure 10.12 React Application index.html code**

**Lab Task(s):** Perform the following task

1. Configure React JS Library and execute the default component of the React JS
2. Create another component of your own name and execute it instead of the default component

## Experiment No. 11 Implementation of React JSX, functional components and props

**Objectives:** To familiarize students with React JSX, functional components and how to pass and receive different props to child components

**Tools:** VS Code, Browser (Internet Explorer, Google Chrome or Firefox)

**Procedure:** To demonstrate as how to return HTML with and without JSX, passing different props from parent to the child components

**JSX (JavaScript XML)** is an extension to JavaScript syntax used by React.js. It allows developers to write HTML-like code directly within JavaScript. JSX makes it easier to describe the structure of UI components and their relationships, making React code more readable and maintainable.

Following are few codes which depicts importance of JSX

```
function JsxTwo() {
   document.write("<h1>I am before JSX");
}
```

The above code doesn't have any JSX code and the HTML part is reflected in the DOM activity. Another code without JSX having the return statement is as follows:

```
function JsxThree() {
   var a = 3;
   var b = 4;
   var c = a + b;
   return c;
}
```

To improve the quality of code having JSX; which will return the whole HTML without any DOM activity can be as follows:

```
function JsxOne() {
  return (
   <div>
      <h1>I am returning JSX</h1>
   </div>
  )
}
```

The **JsxOne** function in the above code is a React component written in JSX syntax. It returns a simple JSX structure consisting of a  containing an <h1> heading with some text. The JSX allows us to write HTML like syntax within a JavaScript, making it easier to describe the UI. The  and <h1> tags resemble

HTML syntax, but they are actually JSX elements. JSX elements are translated into JavaScript function calls by tools like Babel during the build process. When this component is rendered by a parent component or by the ReactDOM, the JsxOne function will be executed. The JSX returned by the function will be rendered as HTML in the browser DOM as shown below:

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import JsxOne from './components/1.JSX';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <JsxOne></JsxOne>
  </React.StrictMode>
);
```

We can include plain text, JavaScript expressions, or other React components within JSX by enclosing them in curly braces {} as shown below:

```
// JS expressions in the HTML, use brackets
const a = 5;    // variable
const obj = {name:"khan",age:"20"};    // object
function JsxOne() {
   return (
      <div>
         <h1>{a}</h1>
         <h1>{2 + 2}</h1>
         <h1>2 + 2</h1>
         <h1>{obj.name}{obj.age}</h1>
      </div>
   )
}
```

Sometimes, we need to add multiple elements in the JS without having an extra node to the DOM. To achieve such things, we need fragments and code can be shown below:

```
function JsxOne () {
   return (
      <>
            <h1>I am using Fragements</h1>
      </>
   )
}
```

The conditional statement can also be used in the React JS. For traditional conditional statements, it is better to use the statements outside the function and access inside the function or use ternary statements to execute directly inside the JSX as shown below:

```
let a = 20;
var result = '';
if(a > 10) {
   result = "Greater"
}
else {
   result = "less";
}
function JsxOne() {
   return (
      <>
         <div>
            <h1>{result}</h1>
            <h1>{ (a > 15)?"ternary true statement":"ternary false statement"}</h1>
         </div>
      </>
   )
}
```

**Functional Components** are a type of component in React that are primarily defined as JavaScript functions. They are also sometimes referred to as stateless functional components or presentational components. In functional components. The following code depicts the functional component example:

```
import React from "react";

function FunctionalComponentstwo() {
   return (
      <>
         <h1>I am functional component</h1>
         <h2>I am second child of the fragments</h2>
      </>
   )

}
export default FunctionalComponentstwo;
```

The above code is a functional component returning a fragment containing two heading elements. The breakdown of the code can be as follow:

- ***import React from "react";*** This imports the React library, which is necessary for writing JSX and creating React components.

- ***function FunctionalComponentstwo() { ... }:*** This declares a functional component named FunctionalComponentstwo using the function keyword. It's a simple function that returns JSX.
- ***return ( ... ):*** Inside the function, we have a return statement that returns a JSX fragment (<></>).
- ***<h1>I am functional component</h1>:*** This is the first child element of the fragment, a <h1> heading element with the text "I am functional component".
- ***<h2>I am second child of the fragments</h2>:*** This is the second child element of the fragment, a <h2> heading element with the text "I am second child of the fragments".
- ***export default FunctionalComponentstwo;:*** Finally, we export the FunctionalComponentstwo function as the default export, allowing it to be imported and used in other files such as src->index.js as shown below:

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import FunctionalComponentstwo from './components/2.functional_components';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <FunctionalComponentstwo>></ FunctionalComponentstwo >
  </React.StrictMode>
);
```

**Props** act as arguments for the functional and class components. Other programming language having arguments or attributes but in React JS, the arguments are called as props. The props are read-only and cannot be changed. In other words, props are used to communicate between the parent and child component or to send any information from the parent to the child component. The following codes depicts the concept of props in traditional JS and React JS

```
// JavaScript function equivalent to Props
function message(msg) {
   document.write(msg);
}
message("hello");
```

The React concept against the traditional JS can be as follows where the props are transferred from the parent component that is src -> index.js

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import PropsConcept from './components/3.Props;

let a = "testing";
const expressions = ["abc","def","ghi"];
```

```
const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <PropsConcept name="Khan" age={20} anyname = {a} data = {expressions}>
       <h4>I am representing the child props and will be accessed with built-in property
            props.children
        </h4>
    </PropsConcept>

  </React.StrictMode>
);
```

The child component code which receives the props are as follows:

```
import React from "react";
function PropsConcept(props) {
   return (
      <>
         <h1>The first prop value is {props.name}</h1>
         <h2>I am second props as {props.age}</h2>
         <h3>I am props received as curly brackets {props.anyname}</h3>
         <h4>{props.children}</h4>
         <h4>{props.data}</h4       </>
   )
}
export default PropsConcept;
```

The breakdown of the above code is as follows:

- **<h1>The first prop value is {props.name}</h1>:** This line renders the value of the name prop passed to the component.
- **<h2>I am second prop as {props.age}</h2>:** Similarly, this line renders the value of the age prop passed to the component.
- **<h3>I am prop received as curly brackets {props.anyname}</h3>:** This line demonstrates how to access a prop with a dynamic name using curly brackets. The actual prop name could be anything, as it's determined by the caller of this component.
- **<h4>{props.children}</h4>:** This line renders the special children prop, which represents any child elements passed to this component.
- **<h4>{props.data}</h4>:** This line demonstrates receiving an array as a prop.

**Lab Task(s):** Perform the following task

3. Create a functional component that renders/display a welcome message to the user.
4. Create a functional component that receives the props and renders your name.
5. Create a functional component that conditionally (use ternary statements) renders contents based on props.
6. Create a list of fruit items and passed as props. The child component should render the fruit items as unordered list.
7. Create a functional component as “ArrowFunction” which receives the props as name and age. The component should render the name and age received as props. The task should be performed using arrow functions.
8. Use your own knowledge to implement the concept of stylesheet in React JS.

## Experiment No. 12 Implementation of class components props, react events

**Objectives:** To familiarize students with React JSX class components with props and react events

**Tools:** VS Code, Browser (Internet Explorer, Google Chrome or Firefox)

**Procedure:** To demonstrate as how to use class components with props

**Class Components**

```
import React, { Component } from 'react';

class Welcome extends Component {
    render() {
        return <h1>Welcome to React, {this.props.name}!</h1>;
    }
}

export default Welcome;
```

In React, a class component is a more traditional way of writing components. It involves extending the `Component` class from React and using the `render` method to return JSX. In the example, the `Welcome` class component displays a welcome message that includes a name passed via props.

**Props with Class Components**

```
import React, { Component } from 'react';

class UserProfile extends Component {
    render() {
        return (
            <div>
                <h2>{this.props.name}</h2>
                <p>Age: {this.props.age}</p>
                <p>Email: {this.props.email}</p>
            </div>
        );
    }
}

export default UserProfile;
```

Props are used to pass data from a parent component to a child component. In class components, props are accessed using `this.props`. In the example, the `UserProfile` class component displays user information such as name, age, and email using props.

**React Events**

```
import React, { Component } from 'react';

class ButtonClick extends Component {
    handleClick() {
        alert('Button was clicked!');
    }

    render() {
        return <button onClick={this.handleClick}>Click Me!</button>;
    }
}

export default ButtonClick;
```

React events work similarly to DOM events in HTML. In class components, event handlers are usually defined as methods and can be passed to elements using the `onClick`, `onChange`, etc., attributes. In the example, a button triggers an alert when clicked.

**Forms in React**

```
import React, { Component } from 'react';

class ContactForm extends Component {
    constructor(props) {
        super(props);
        this.state = { name: '', email: '' };

        this.handleChange = this.handleChange.bind(this);
        this.handleSubmit = this.handleSubmit.bind(this);
    }

    handleChange(event) {
        const { name, value } = event.target;
        this.setState({ [name]: value });
    }

    handleSubmit(event) {
        event.preventDefault();
        alert(`Name: ${this.state.name}, Email: ${this.state.email}`);
    }

    render() {
        return (
            <form onSubmit={this.handleSubmit}>
                <label>
                    Name:
                    <input type="text" name="name" value={this.state.name} onChang
                </label>
                <br />
                <label>
                    Email:
                    <input type="email" name="email" value={this.state.email} onCh
                </label>
                <br />
                <button type="submit">Submit</button>
            </form>
        );
    }
}

export default ContactForm;
```

Forms in React are handled using controlled components, where form data is managed by the component's state. In a class component, the form input values are stored in the component's state and updated via the `handleChange` method. The form submission is managed by the `handleSubmit` method, which can also handle any necessary validations or operations. In the example, a simple contact form captures the user's name and email.

**Lab Tasks: Perform the following**

1. Create a simple Welcome class component that displays a "Welcome to React" message.

2. Create a UserGreeting class component that takes a name prop and displays a personalized greeting, e.g., "Hello, [name]!".
3. Create a new class component named `ButtonClick`. Inside `ButtonClick`, render a button with the text "Click Me".Implement an event handler that show alert "Clicked!" when it is clicked.

Create a new class component named `UserForm`. Inside `UserForm`, render a form with two input fields: one for the user's name and another for their email. Implement `onChange` handlers for both input fields to update the component's state with the entered values. Add a submit button, and when the form is submitted, log the entered name and email to the

## Experiment No. 13 Implementation of react hooks

**Objectives:** To familiarize students with React JSX hooks

**Tools:** VS Code, Browser (Internet Explorer, Google Chrome or Firefox)

**Procedure:** To demonstrate as how to use hooks in the React JSX

**UseState**

```
import React, { useState } from 'react';

function Counter() {
    const [count, setCount] = useState(0);

    return (
        <div>
            <p>You clicked {count} times</p>
            <button onClick={() => setCount(count + 1)}>
                Click me
            </button>
        </div>
    );
}

export default Counter;
```

The `useState` hook is a fundamental hook in React that allows you to add state to functional components. In the example above, `useState(0)` initializes the `count` state variable to `0`. The `setCount` function is used to update this state.
When the button is clicked, `setCount` updates the `count`, causing the component to re-render with the new state value.

**useEffect**

```
import React, { useState, useEffect } from 'react';

function Timer() {
    const [count, setCount] = useState(0);

    useEffect(() => {
        const timer = setInterval(() => {
            setCount(count + 1);
        }, 1000);

        return () => clearInterval(timer);
    }, [count]);

    return <h1>{count} seconds passed</h1>;
}

export default Timer;
```

The `useEffect` hook is used to perform side effects in functional components, such as data fetching, subscriptions, or manually changing the DOM.
In the example above, `useEffect` is used to start a timer that updates the `count` state every second. The return function inside `useEffect` cleans up the timer to avoid memory leaks, ensuring that the

interval is cleared when the component unmounts or before the next effect runs.

**useContext**

```
import React, { useContext } from 'react';

const UserContext = React.createContext();

function DisplayUser() {
    const user = useContext(UserContext);

    return <h1>{user.name}</h1>;
}

function App() {
    const user = { name: 'John Doe' };

    return (
        <UserContext.Provider value={user}>
            <DisplayUser />
        </UserContext.Provider>
    );
}

export default App;
```

The `useContext` hook allows functional components to subscribe to context changes.
In this example, `UserContext` is created using `React.createContext`.
The `App` component provides a `user` object as the context value, and the `DisplayUser` component consumes this context using `useContext(UserContext)`.
This allows the `DisplayUser` component to access the `user` object and render the user's name.

**useReducer**

```
import React, { useReducer } from 'react';

function reducer(state, action) {
    switch (action.type) {
        case 'increment':
            return { count: state.count + 1 };
        case 'decrement':
            return { count: state.count - 1 };
        default:
            throw new Error();
    }
}

function Counter() {
    const [state, dispatch] = useReducer(reducer, { count: 0 });

    return (
        <div>
            <p>Count: {state.count}</p>
            <button onClick={() => dispatch({ type: 'increment' })}>
                Increment
            </button>
            <button onClick={() => dispatch({ type: 'decrement' })}>
                Decrement
            </button>
        </div>
    );
}

export default Counter;
```

The `useReducer` hook is an alternative to `useState` for managing more complex state logic. It accepts a reducer function and an initial state as arguments, returning the current state and a dispatch function.

In this example, the `reducer` function handles `increment` and `decrement` actions, and `dispatch` is used to trigger state updates based on these actions.

The `Counter` component uses `useReducer` to maintain and update the `count` state based on dispatched actions.

**useRef**

```
import React, { useRef } from 'react';

function FocusInput() {
    const inputEl = useRef(null);

    const onButtonClick = () => {
        inputEl.current.focus();
    };

    return (
        <div>
            <input ref={inputEl} type="text" />
            <button onClick={onButtonClick}>Focus the input</button>
        </div>
    );
}

export default FocusInput;
```

The `useRef` hook provides a way to access DOM elements directly or keep a mutable reference that persists across re-renders.
In the example above, `useRef` is used to create a reference (`inputEl`) to an input element. The `onButtonClick` function uses this reference to call the `focus` method on the input element, giving it focus when the button is clicked.
`useRef` is also commonly used to store any mutable value that doesn't cause a re-render when changed.

**Lab Tasks: Perform the following:**

1. Create a counter component that displays a number and has two buttons: "Increment" and "Decrement". Use the useState hook.
2. Create a component with a button labeled "Toggle".Clicking the button should show or hide a paragraph of text.
3. Create a component that tracks the number of times a button is clicked. Use useEffect to update the document title with the count each time the count changes.
4. Create a ThemeContext with two themes: light and dark. Create two components: one for displaying the theme and another for toggling between themes. Use useContext to access the current theme and apply it to the components.
5. Create a counter component similar to the useState example but use useReducer. Define actions for increment, decrement, and reset.
6. Create a form with an input field. Use useRef to automatically focus the input field when the component loads.

## Experiment No. 14 Implementation of react JSX routing & navigator

**Objectives:** To familiarize students with React JSX routing and page shifting

**Tools:** VS Code, Browser (Internet Explorer, Google Chrome or Firefox)

**Procedure:** To demonstrate as how to visit different pages

**React JSX**

```
import React from 'react';

function Greeting() {
    const name = 'John Doe';
    return <h1>Hello, {name}!</h1>;
}

export default Greeting;
```

**React Router Setup**

```
import React from 'react';
import { BrowserRouter as Router, Routes, Route } from 'react-router-dom';
import Home from './Home';
import About from './About';

function App() {
    return (
        <Router>
            <Routes>
                <Route path="/" element={<Home />} />
                <Route path="/about" element={<About />} />
            </Routes>
        </Router>
    );
}

export default App;
```

React Router is a library for managing navigation in a React application. The example above shows the basic setup using `BrowserRouter` as `Router`, `Routes`, and `Route`. `Route` elements define the paths and corresponding components. When the user navigates to `/`, the `Home` component is rendered, and when they navigate to `/about`, the `About` component is rendered.

**Navigation Between Routes**

```
import React from 'react';
import { Link, useNavigate } from 'react-router-dom';

function Navbar() {
    const navigate = useNavigate();

    return (
        <nav>
            <ul>
                <li><Link to="/">Home</Link></li>
                <li><Link to="/about">About</Link></li>
                <li><button onClick={() => navigate('/contact')}>Contact</button><
            </ul>
        </nav>
    );
}

export default Navbar;
```

React Router provides components like `Link` and hooks like `useNavigate` to facilitate navigation between different routes.
`Link` is used to create anchor links that navigate to a different route without reloading the page. `useNavigate` is a hook that allows programmatic navigation, like redirecting users after a form submission or button click. In the example, `Link` is used to navigate to the "Home" and "About" pages, while `useNavigate` is used to navigate to the "Contact" page programmatically when the button is clicked.

**Lab Tasks: Perform the following:**

1. Create three components: `Home`, `About`, and `Contact`. Set up routing so that navigating to / displays the Home component, /about displays the About component, and /contact displays the Contact component.
2. Create a Product component that displays product details based on a dynamic productId parameter. Set up a route like /product/:productId .Extract the param and display it.
3. Create a login form that, upon successful submission, redirects the user to a `Dashboard` component. Use the useNavigate hook to navigate to the Dashboard programmatically after the form is submitted.

## Experiment No. 15 Installation of Wordpress, Dashboard Review

**Objectives:** To familiarize students with the basics of Wordpress

**Tools:** XAMPP Server, Dreamweaver, Browser (Internet Explorer, Google Chrome or Firefox)

**Procedure:** Installation of Wordpress and theme upload.

WordPress is an open source **Content Management System (CMS),** which allows the users to build dynamic websites and blogs. WordPress is the most popular blogging system on the web and allows updating, customizing and managing the website from its back-end CMS and components.

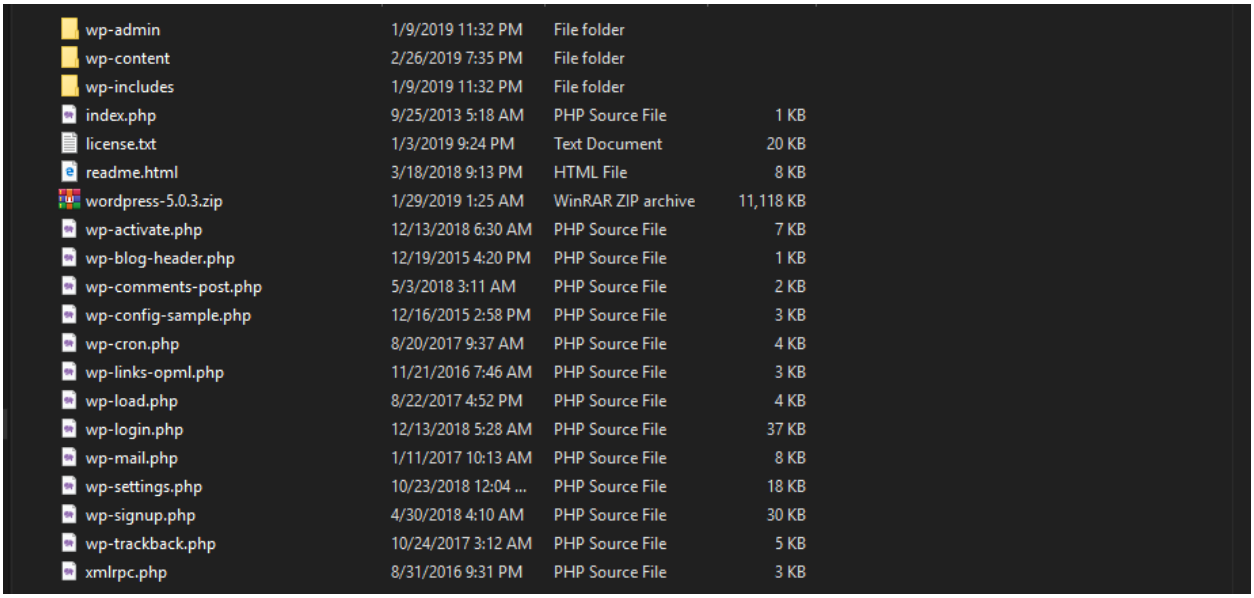
The **Content Management System (CMS)** is a software which stores all the data such as text, photos, music, documents, etc. and is made available on your website. It helps in editing, publishing and modifying the content of the website. The different features of the Wordpress are:

- **User Management** − It allows managing the user information such as changing the role of the users to (subscriber, contributor, author, editor or administrator), create or delete the user, change the password and user information. The main role of the user manager is Authentication.
- **Media Management** − It is the tool for managing the media files and folder, in which you can easily upload, organize and manage the media files on your website.
- **Theme System** − It allows modifying the site view and functionality. It includes images, stylesheet, template files and custom pages.
- **Extend with Plugins** − Several plugins are available which provides custom functions and features according to the users need.
- **Search Engine Optimization** − It provides several search engine optimization (SEO) tools which makes on-site SEO simple.
- **Multilingual –** It allows translating the entire content into the language preferred by the user.
- **Importers** − It allows importing data in the form of posts. It imports custom files, comments, post pages and tags.

**Step 1**. Download latest version of wordpress from: **https://wordpress.org/download/**

**Step 2.** Download and Install XAMPP local server in your computer and start the apache and MYSQL server.

**Step 3.** Extract the Wordpress zip file and copy the contents to: **C://xampp/htdocs**



| | | | |
|---|---|---|---|
| wp-admin | 1/9/2019 11:32 PM | File folder | |
| wp-content | 2/26/2019 7:35 PM | File folder | |
| wp-includes | 1/9/2019 11:32 PM | File folder | |
| index.php | 9/25/2013 5:18 AM | PHP Source File | 1 KB |
| license.txt | 1/3/2019 9:24 PM | Text Document | 20 KB |
| readme.html | 3/18/2018 9:13 PM | HTML File | 8 KB |
| wordpress-5.0.3.zip | 1/29/2019 1:25 AM | WinRAR ZIP archive | 11,118 KB |
| wp-activate.php | 12/13/2018 6:30 AM | PHP Source File | 7 KB |
| wp-blog-header.php | 12/19/2015 4:20 PM | PHP Source File | 1 KB |
| wp-comments-post.php | 5/3/2018 3:11 AM | PHP Source File | 2 KB |
| wp-config-sample.php | 12/16/2015 2:58 PM | PHP Source File | 3 KB |
| wp-cron.php | 8/20/2017 9:37 AM | PHP Source File | 4 KB |
| wp-links-opml.php | 11/21/2016 7:46 AM | PHP Source File | 3 KB |
| wp-load.php | 8/22/2017 4:52 PM | PHP Source File | 4 KB |
| wp-login.php | 12/13/2018 5:28 AM | PHP Source File | 37 KB |
| wp-mail.php | 1/11/2017 10:13 AM | PHP Source File | 8 KB |
| wp-settings.php | 10/23/2018 12:04 ... | PHP Source File | 18 KB |
| wp-signup.php | 4/30/2018 4:10 AM | PHP Source File | 30 KB |
| wp-trackback.php | 10/24/2017 3:12 AM | PHP Source File | 5 KB |
| xmlrpc.php | 8/31/2016 9:31 PM | PHP Source File | 3 KB |

Figure 15.1: Wordpress Extraction

**Step 4**. Create Database for your Wordpress Site through PHPMYADMIN.


Figure 15.2: Wordpress Database

**Step 5.** Open Wordpress package in your browser by typing: **http://localhost/wordpress**

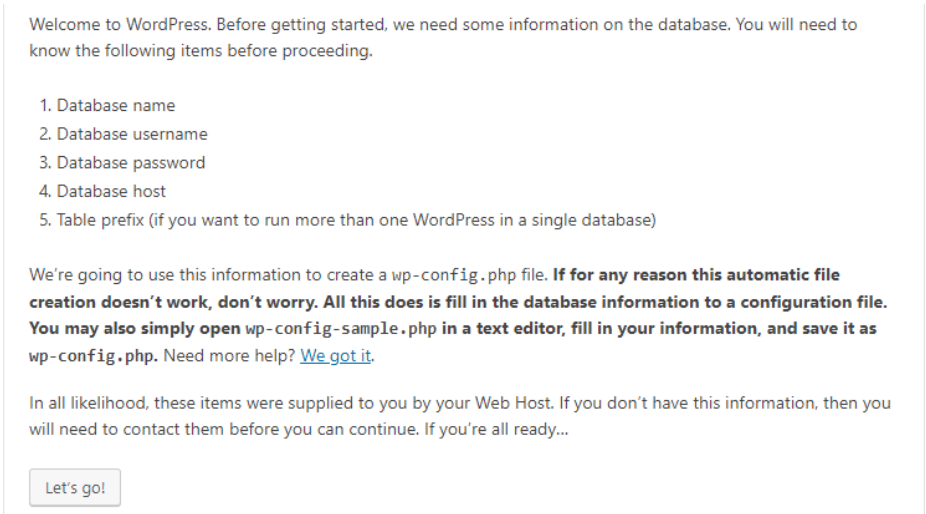



Figure 15.3: Wordpress Extraction

**Step 6**. Click on Lets Go button to begin the installation of wordpress.

**Step 7.** Configure the fields according to your database. For local host, these configurations are:

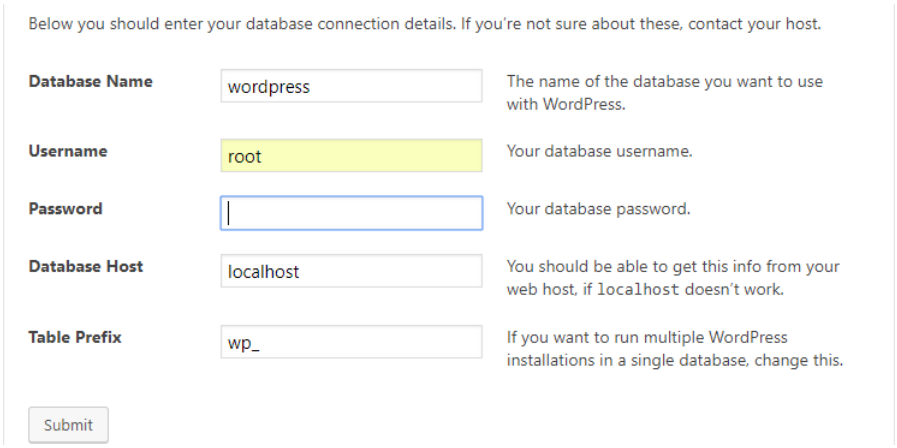



Figure 15.4: Wordpress Configuration

**Step 8.** Fill all the required fields in welcome page with information required.

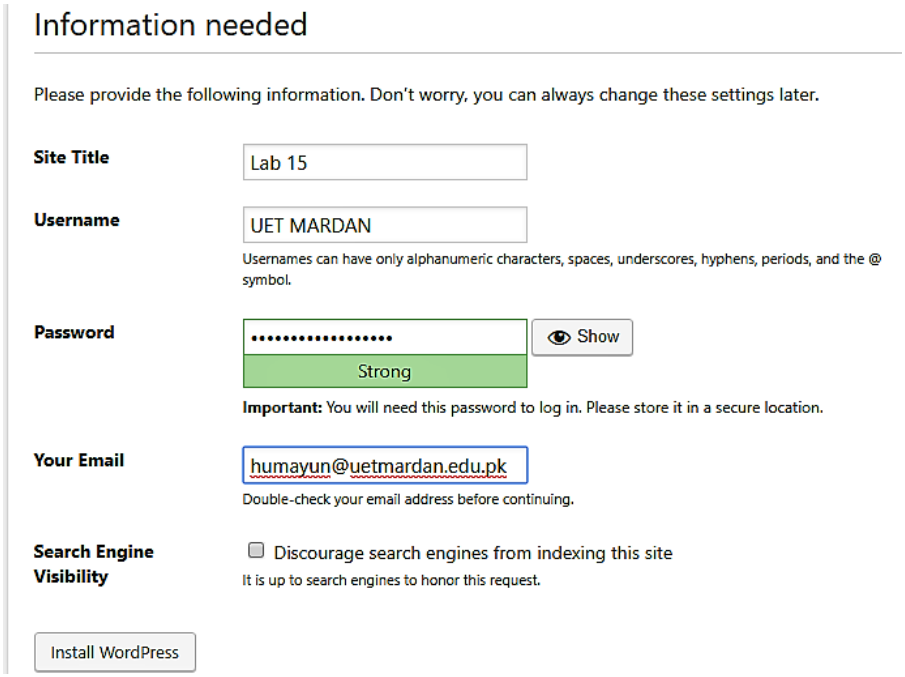



Figure 15.5: Wordpress information

**Step 9.** Run and Install the Wordpress.

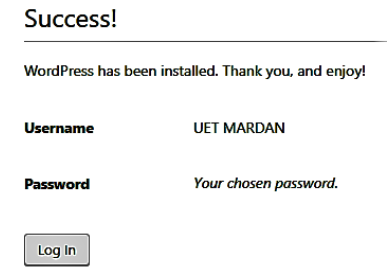



Figure 15.6: Wordpress Login Details

**Step 10.** Log in to your Admin Panel by going to the URL: **http://localhost/wordpress/wp-admin**

**Step 11.** Login to your Dash Board with the username and password used for creating website.

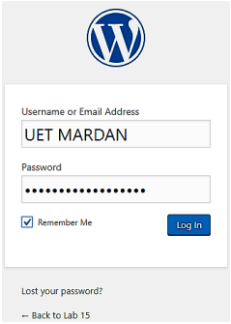



Figure 15.7: Wordpress Login

# Dashboard Review

The dashboard of the Wordpress after installation is shown in Figure 68

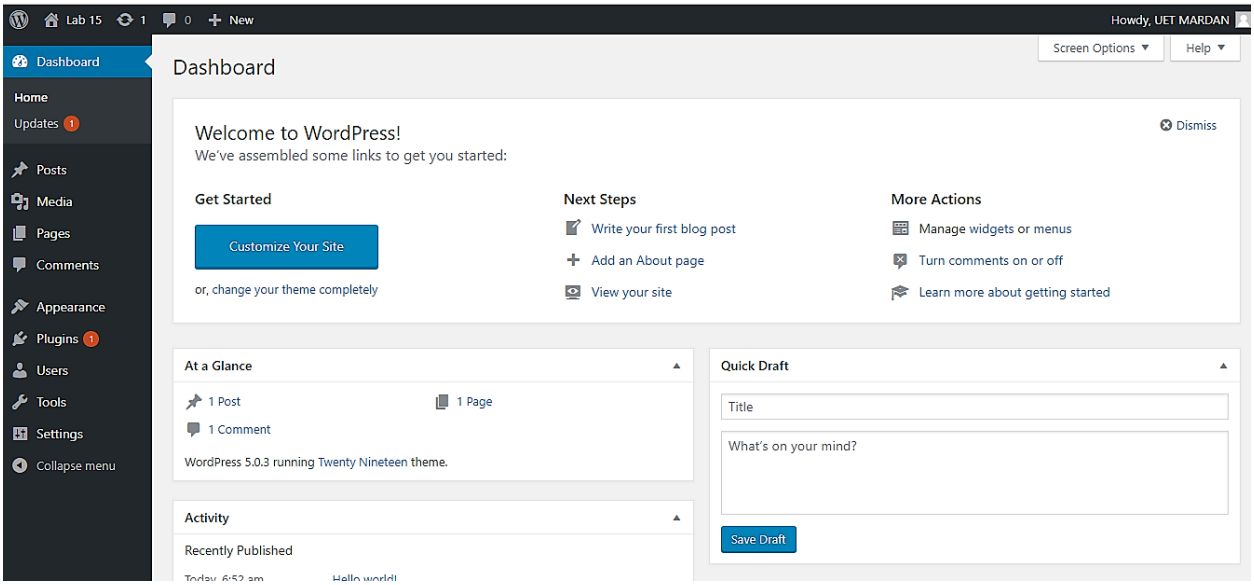



Figure 15.8: Wordpress Dashboard

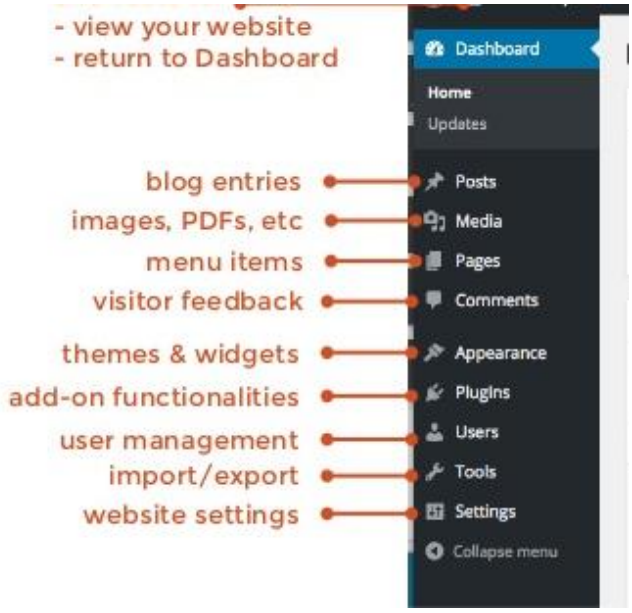



Figure 15.9: Dashboard Menu

The dashboard Posts menu item is used to create new blog post. From here, we can also update our categories and Past Tags. The posts menu item is shown in Figure 70 below.

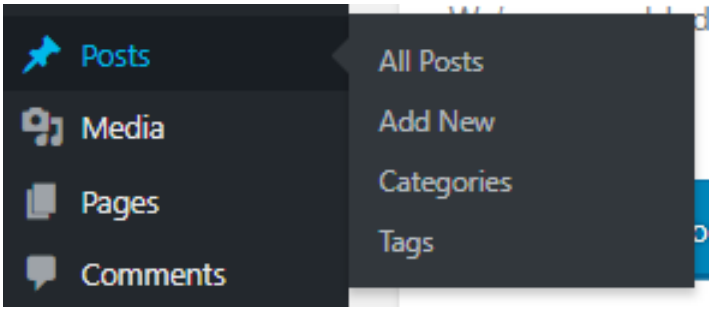



Figure 15.10: Dashboard Posts Menu Item

The media menu item is used to upload images, documents or files. It can also browse through media library, as well as edit and update the files. The media menu item is shown in Figure 71 below.

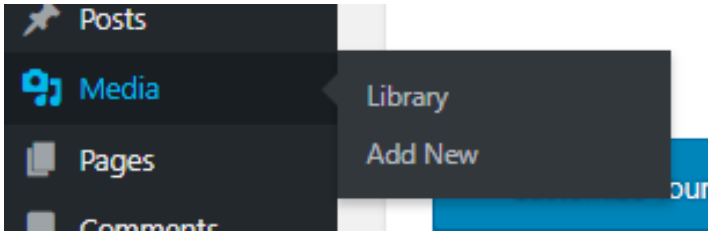



Figure 15.11: Dashboard Media Menu Item

The pages menu item is used to create and maintain all your pages. All the content of website which changes with time is defined in pages. The pages menu item is shown in Figure 72 below.

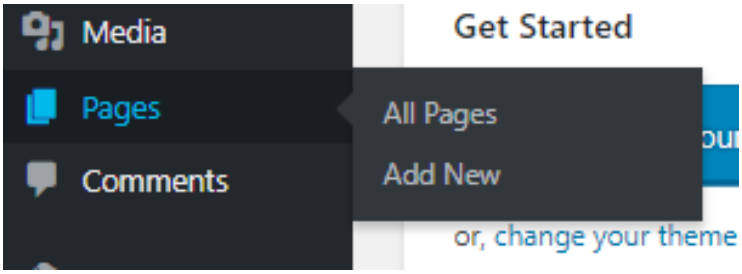



Figure 15.12: Dashboard Pages Menu Item

The commens menu item manages all comments within this section, including replying to comments or marking them as spam. The comments menu item is shown in Figure 73 below.

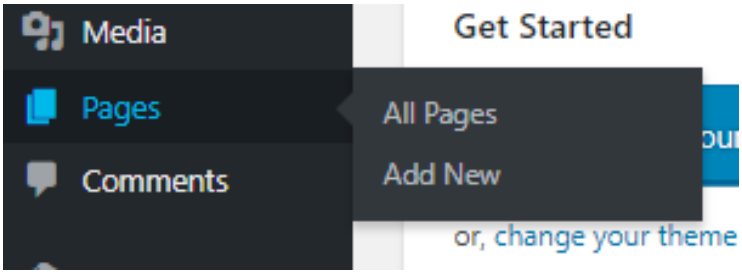



Figure 15.13: Dashboard Comments Menu Item

The appearance menu item is used to control how your site looks. You can choose a new Theme, manage your site Widgets or Menus and even edit your site theme files. The Figure 74 is for appearance menu item.

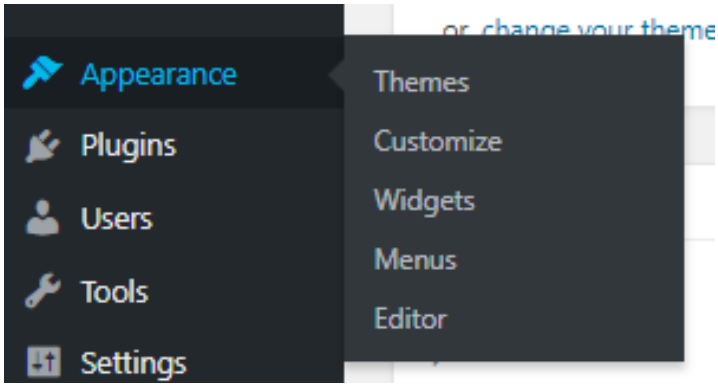



Figure 15.14: Dashboard Appearance Menu Item

The plugins menu item extends and expand the functionality of the website. We can add the plugins, delete, activate or deactivate. The Figure 75 is for plugins menu item.

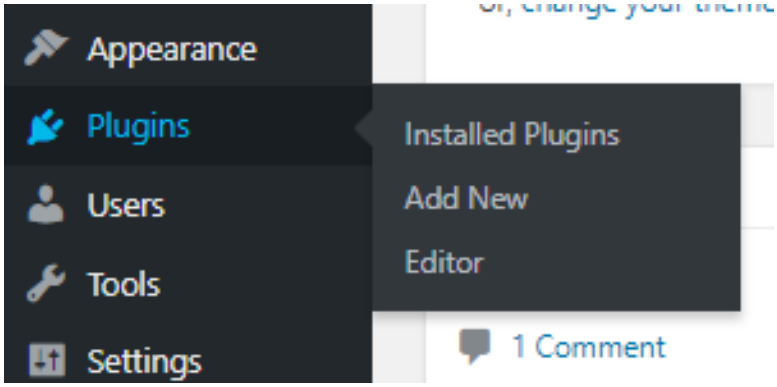



Figure 15.15: Dashboard Plugins Menu Item

The users menu item lists all the existing users for the site. Depending upon the role, we can also add new users as well as manage their roles. The different users can be as administrator (has right to do anything), editor (can manage and publish post and pages of hemself as well as of others), author (Can manage and publish posts and pages of hemself), contributor (write posts but cannot publish them), subscriber (view posts). The Figure 76 is for users menu item.

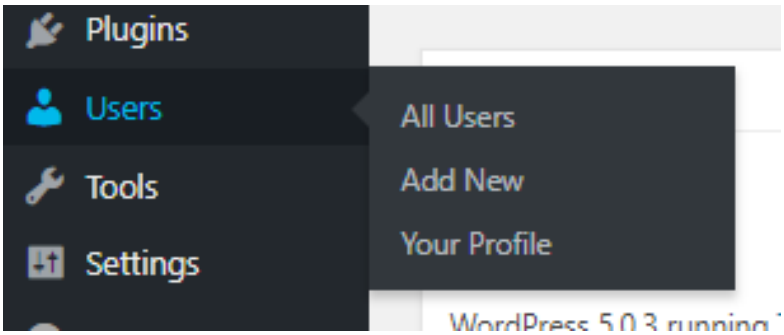



Figure 15.16: Dashboard Users Menu Item

The tools menu item gives access to various convenient tools and also import data to Wordpress site or export Wordpress data to a file. The Figure 77 is for tools menu item.

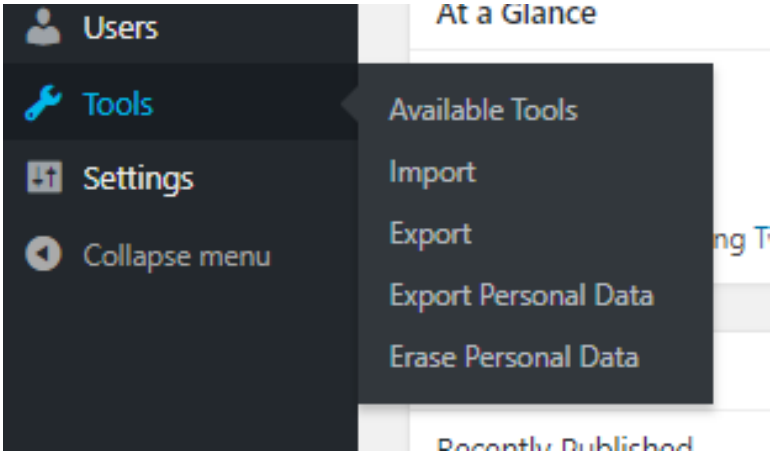



Figure 15.17: Dashboard Tools Menu Item

The settings menu item configured the website. Among other things, it allows you to configure site name and URL, where posts appear, whether people can leave comments or not and numeours other settings. The Figure 78 is for settings menu item.

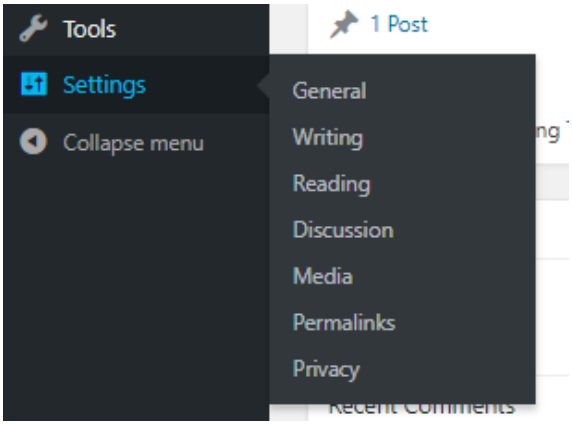
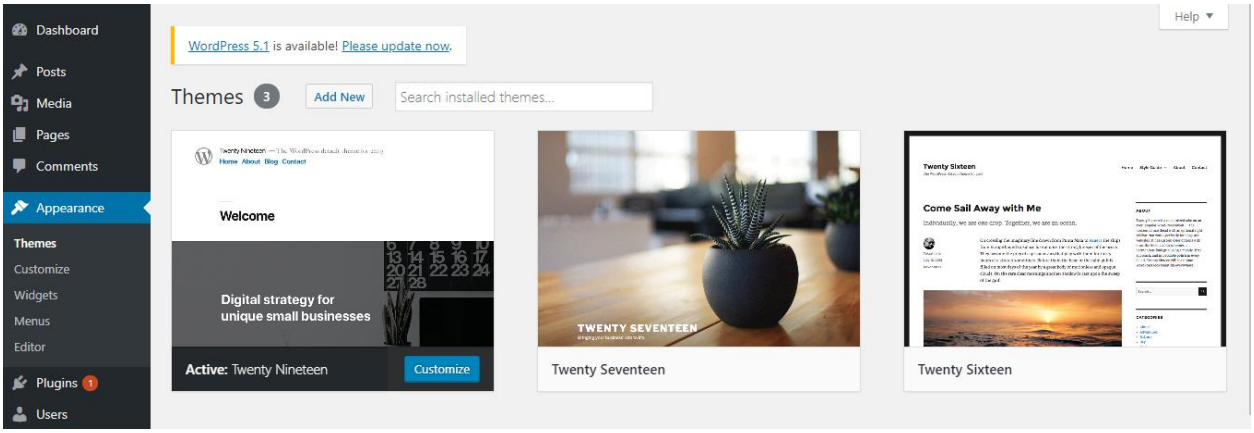



Figure 15.18: Dashboard Setting Menu Item

# Uploading Theme in Wordpress

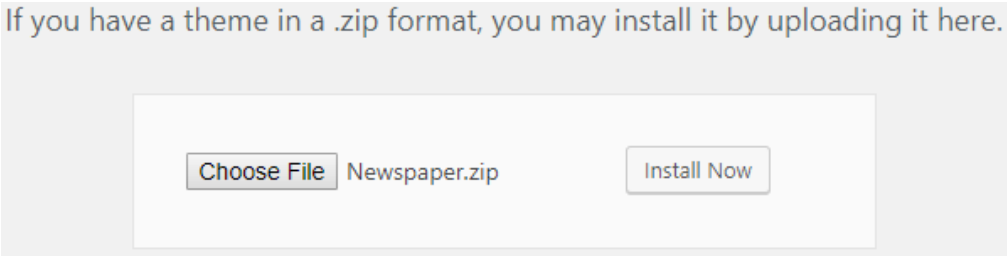
**Method 1:**

**Step 1.** From Wordpress Dashboard, goto to **Appereance** menu and click **Theme**




**Step 2**. Click on Add New button and install theme from WordPress theme store or Click on Upload Theme button.

**Step 3**. Click on Choose File button and browse for the theme zip file.




**Step 4.** Click on Install Now to Install the theme.

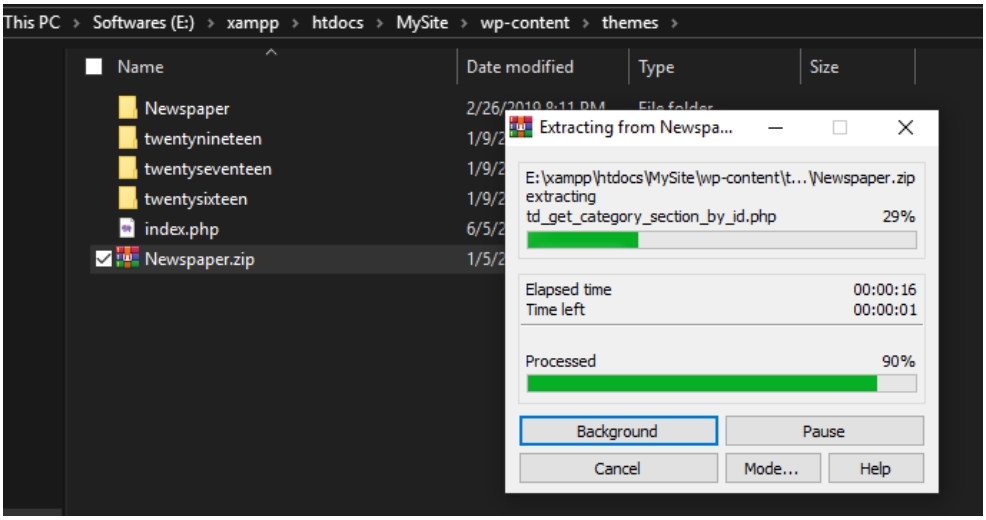
**Step 5**. You can Activate the theme through Appearance -> Theme menu.

# Method 2:

**Step 1.** Download the theme Zip File from 3rd party theme providing stores.

**Step 2**. Extract the theme zip file and paste the extracted files into:

**C://xampp/htdocs/wordpress/wp-content/themes**




**Step 3**. Open your Dashboard and and goto Appearance -> Theme

**Step 4**. Activate The installed theme.

**Lab Task(s):** Perform the following:

1. Install Wordpress environment on your own machine.
2. Install a Wordpress theme from the dashboard.
3. Upload a Wordpress theme from a zip file.