

**HariOm Shukla****EXPERIMENT – 01****121A3052**

---

**Aim:** To create a registration page using HTML tags.

**THEORY:**

**HTML:**

The Hyper Text Markup Language, or HTML is the standard markup language for documents designed to be displayed in a web browser. It can be assisted by technologies such as Cascading Style Sheets (CSS) and scripting languages such as JavaScript. Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document.

**HTML Tags:**

HTML tags are like keywords which define how web browser will format and display the content. With the help of tags, a web browser can distinguish between HTML content and simple content. HTML tags contain three main parts: opening tag, content and closing tag. But some HTML tags are unclosed tags. When a web browser reads an HTML document, the browser reads it from top to bottom and left to right. HTML tags are used to create HTML documents and render their properties. Each HTML tag has different properties. An HTML file must have some essential tags so that web browser can differentiate between a simple text and HTML text. You can use as many tags as you want as per your code requirement.

- All HTML tags must be enclosed within < > these brackets.
- Every tag in HTML performs different tasks.
- If you have used an open tag, then you must use a close tag (except some tags)

The syntax for a tag is: content. Some examples for tags in HTML are:

<p> Paragraph Tag </p>

**<h2> Heading Tag </h2>**

<b> **Bold Tag** </b>

<i> *Italic Tag* </i>

<u> Underline Tag </u>

```
<!doctype html>
<head>
<title>
Student Registration Form
</title>
</head>
<body>
<div class="container">
<form>
<h1 align = center>Student Registration Form</h1>
<h2>NAME:</h2>
<input type = "text" placeholder = "Enter Name">
<h2>ROLL NO.:</h2>
<input type = "text" placeholder = "Enter Roll No.">
<h2>USERNAME:</h2>
<input type = "text" placeholder = "Enter Username">
<h2>PASSWORD</h2>
<input type = "password" placeholder = "Enter Password">
<input type="submit" value="SUBMIT">
</form>
</div>
</body>
<style>
input[type=text], input[type=password]{
width: 100%;
padding: 15px;
box-sizing: border-box;
border: 2px solid;
}
input[type=submit] {
margin-top: 27px;
padding: 18px 24px;
cursor: pointer;
background-color: #F90097;
}
.container {
background-image: url("img.jpeg");
min-height: 611px;
background-position: center;
background-repeat: no-repeat;
background-size: cover;
position: relative;
padding: 20px;
}
```

</style>  
</html>

**Student Registration Form**

**NAME:**

HariOm

**ROLL NO.:**

121A3052

**USERNAME:**

hariom

**PASSWORD**

\*\*\*\*\*

SUBMIT

Activate Windows  
Go to Settings to activate Windows.

**Student Registration Form**

**NAME:**

Enter Name

**ROLL NO.:**

Enter Roll No.

**USERNAME:**

Enter Username

**PASSWORD**

Enter Password

SUBMIT

Activate Windows  
Go to Settings to activate Windows.

**CONCLUSION:** Through this experiment we learned how to create a registration form using html tags.

**HariOm Shukla****EXPERIMENT – 02****121A3052**

---

**Aim:** To create a navigation bar using HTML and CSS.

**THEORY:**

**HTML:**

The Hyper Text Markup Language, or HTML is the standard markup language for documents designed to be displayed in a web browser. It can be assisted by technologies such as Cascading Style Sheets (CSS) and scripting languages such as JavaScript. Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document.

**CSS:**

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language such as HTML. CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript. CSS is designed to enable the separation of presentation and content, including layout, colours, and fonts.

This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple web pages to share formatting by specifying the relevant CSS in a separate .css file which reduces complexity and repetition in the structural content as well as enabling the .css file to be cached to improve the page load speed between the pages that share the file and its formatting.

The name cascading comes from the specified priority scheme to determine which style rule applies if more than one rule matches a particular element. This cascading priority scheme is predictable. Separation of formatting and content also makes it feasible to present the same markup page in different styles for different rendering methods, such as on-screen, in print, by voice (via speech-based browser or screen reader), and on Braille-based tactile devices. CSS also has rules for alternate formatting if the content is accessed on a mobile device.

The CSS specifications are maintained by the World Wide Web Consortium (W3C). Internet media type (MIME type) text/css is registered for use with CSS by RFC 2318 (March 1998).

The W3C operates a free CSS validation service for CSS documents

In addition to HTML, other markup languages support the use of CSS including XHTML, plain XML, SVG, and XUL.

```
<!doctype html>
<head>
<title>
NAVIGATION BAR
</title>
</head>
<body>
<div class = "topnav">
<a class = active>HOME</a>
<a>ABOUT</a>
<a>SERVICES</a>
<a>CONTACT US</a>
</div>
</body>
<style>
.topnav {
overflow: hidden;
min-height: 952px;
background-position: center;
background-repeat: no-repeat;
background-size: cover;
}
.topnav a {
float: left;
color: blue;
text-align: center;
padding: 25px 29px;
text-decoration: none;
font-size: 21px;
}
/* Change the color of links on hover */
.topnav a:hover {
background-color: #ede91b;
color: black;
}
```

HOME

ABOUT

SERVICES

CONTACT US

**CONCLUSION:** We have successfully completed navigation bar using HTML and CSS.

**HariOm Shukla****EXPERIMENT – 03****121A3052**

**Aim:** To create a login page using HTML and CSS.

**THEORY:**

**HTML:**

The Hyper Text Markup Language, or HTML is the standard markup language for documents designed to be displayed in a web browser. It can be assisted by technologies such as Cascading Style Sheets (CSS) and scripting languages such as JavaScript. Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document.

**CSS:**

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language such as HTML. CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript. CSS is designed to enable the separation of presentation and content, including layout, colours, and fonts. This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple web pages to share formatting by specifying the relevant CSS in a separate .css file which reduces complexity and repetition in the structural content as well as enabling the .css file to be cached to improve the page load speed between the pages that share the file and its formatting.

The name cascading comes from the specified priority scheme to determine which style rule applies if more than one rule matches a particular element. This cascading priority scheme is predictable. Separation of formatting and content also makes it feasible to present the same markup page in different styles for different rendering methods, such as on-screen, in print, by voice (via speech-based browser or screen reader), and on Braille-based tactile devices. CSS also has rules for alternate formatting if the content is accessed on a mobile device. The CSS specifications are maintained by the World Wide Web Consortium (W3C).

Internet media type (MIME type) text/css is registered for use with CSS by RFC 2318 (March 1998). The W3C operates a free CSS validation service for CSS documents. In addition to HTML, other markup languages support the use of CSS including XHTML, plain XML, SVG, and XUL.

**Code:**

```
<!DOCTYPE html>
<html>
<head>
  <title>Registration Form</title>
</head>
<body>
  <link rel="stylesheet" href="style.css">

  <div class="container">
    <h1>Register Here</h1>
    <p>Please fill in this form to create an account.</p><hr>
```

<label for="name"><b>Name</b></label>

<input type="text" placeholder="Enter Your Name" name="name" id="name" required>

<label for="rollno"><b>Roll-No</b></label>

<input type="text" placeholder="Enter Your RollNo" name="rollno" id="rollno" required>

<label for="email"><b>Email</b></label>

<input type="text" placeholder="Enter Your Email" name="email" id="email" required>

<label for="pwd">Password:</label>

<input type="password" id="pwd" name="pwd" minlength="8"><br><br>

<label for="psw-repeat"><b>Repeat Password</b></label>

<input type="password" id="pwd" name="pwd" minlength="8"><br><br>

<label for="department"><b>Department</b></label>

<select name="depart" id="depart">

<option value="Information Technology">Information Technology</option>

<option value="Computer">Computer</option>

<option value="Civil">Civil</option>

<option value="Mechanical">Mechanical</option>

<option value="AIML">AIML</option>

<option value="AIDS">AIDS</option>

</select>

<p><b>Please select your Gender:</b></p>

<input type="radio" name="gender" value="male" checked> Male<br>

<input type="radio" name="gender" value="female"> Female<br>

<input type="radio" name="gender" value="other"> Other

<p><b>Please select Area Of Interest:</b></p>

<input type="checkbox" id="typ1" name="vehicle1" value="Programming">

<label for="typ1"> Programming</label><br>

<input type="checkbox" id="typ2" name="vehicle2" value="Logistics">

<label for="typ2"> Logistics</label><br>

<input type="checkbox" id="typ3" name="vehicle1" value="Data Analyst">

<label for="vehicle1"> Data Analyst</label><br>

<input type="checkbox" id="typ4" name="vehicle3" value="Marketing">

<label for="typ4"> Marketing</label><br><br>

<input type="submit" value="Submit">

<hr>

</body>

</html>



## Register Here

Please fill in this form to create an account.

Name  Roll-No  Email  Password:

Repeat Password

Department

Please select your Gender:

- ☒ Male  
☐ Female  
☐ Other

Please select Area Of Interest:

- ☒ Programming  
☐ Logistics  
☒ Data Analyst  
☐ Marketing

**CONCLUSION:** We have successfully completed login page using HTML and CSS.

**HariOm Shukla****EXPERIMENT – 04****121A3052**

**Aim:** To create a web page using Bootstrap.

**THEORY:****HTML:**

The Hyper Text Markup Language, or HTML is the standard markup language for documents designed to be displayed in a web browser. It can be assisted by technologies such as Cascading Style Sheets (CSS) and scripting languages such as JavaScript. Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document.

**Bootstrap:**

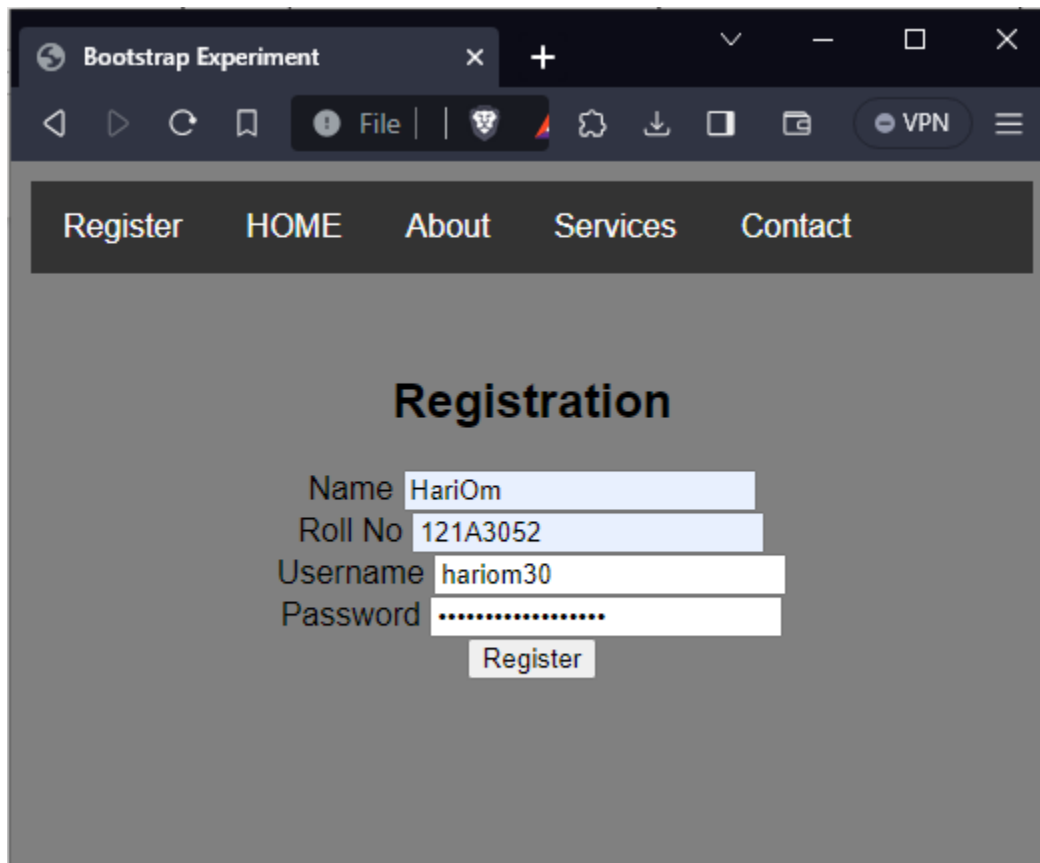
Bootstrap is a free and open-source front-end development framework designed for responsive and mobile-first web development. It provides a collection of syntax for template designs, enabling developers to build websites faster without worrying about basic commands and functions. Bootstrap contains HTML, CSS, and JavaScript-based design templates for typography, forms, buttons, navigation, and other interface components. It is considered one of the most widely used HTML, CSS, and JavaScript frameworks for creating mobile-friendly and responsive websites.<sup>3</sup> Bootstrap Icons is an open-source SVG icon library featuring over 1,800 glyphs, with more added every release.

**Code:**

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Bootstrap Experiment</title>
  <style>
    .registration {
      margin-bottom: 10px;
      padding: 30px;
      text-align: center;
      border: #111;
    }
    body {
      align-items: center;
      margin: 0;
      font-family: Arial, Helvetica, sans-serif;
      padding: 10px;
    }
  </style>
</head>
<body>
```

```
ul.navbar {
    list-style-type: none;
    margin: 0;
    padding: 0;
    background-color: #333;
    overflow: hidden;
}
li.nav-item {
    float: left;
}
li.nav-item a {
    display: block;
    color: white;
    text-align: center;
    padding: 14px 16px;
    text-decoration: none;
}
li.nav-item a:hover {
    background-color: #111;
}
</style>
</head>
<body bgcolor="grey">
<ul class="navbar">
    <li class="nav-item"><a href="registration.html">Register</a></li>
    <li class="nav-item"><a href="home.html">HOME</a></li>
    <li class="nav-item"><a href="construction.html">About</a></li>
    <li class="nav-item"><a href="construction.html">Services</a></li>
    <li class="nav-item"><a href="construction.html">Contact</a></li>
</ul>
<div class="container" style="border-radius: 10px;">
    <div class="registration">
        <form action="#" method="POST" class="registration-form">
            <h2>Registration</h2>
            <div>
                <label for="name">Name</label>
                <input type="text" id="name" name="name" required>
            </div>
            <div>
                <label for="rollno">Roll No</label>
                <input type="text" id="rollno" name="rollno" required>
            </div>
            <div>
                <label for="username">Username</label>
```

```
<input type="text" id="username" name="username" required>
</div>
<div>
  <label for="password">Password</label>
  <input type="password" id="password" name="password" required>
</div>
<input type="submit" value="Register">
</form>
</div>
</div>
</body>
</html>
```

**Output:**

The screenshot shows a web browser window titled "Bootstrap Experiment". The browser's address bar shows "File" and a "VPN" button. The webpage has a dark navigation bar with links: "Register", "HOME", "About", "Services", and "Contact". The main content area has a light gray background and is titled "Registration" in bold. Below the title, there are four input fields: "Name" with the value "HariOm", "Roll No" with the value "121A3052", "Username" with the value "hariom30", and "Password" with masked characters ".....". A "Register" button is positioned below the password field.

**CONCLUSION:** Through this experiment we learned how to create a webpage using Bootstrap.

**HariOm Shukla****EXPERIMENT – 05****121A3052**

---

**Aim:** To create a webpage with any 5 events using html & javascript.

**THEORY:****HTML:**

The Hyper Text Markup Language, or HTML is the standard markup language for documents designed to be displayed in a web browser. It can be assisted by technologies such as Cascading Style Sheets (CSS) and scripting languages such as JavaScript. Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document.

**JavaScript:**

JavaScript (JS) is a lightweight interpreted (or just-in-time compiled) programming language with first-class functions. While it is most well-known as the scripting language for Web pages, many non-browser environments also use it, such as Node.js, Apache CouchDB and Adobe Acrobat. JavaScript is a prototype-based, multi-paradigm, single-threaded, dynamic language, supporting object-oriented, imperative, and declarative (e.g. functional programming) styles.

JavaScript's dynamic capabilities include runtime object construction, variable parameter lists, function variables, dynamic script creation (via eval), object introspection (via for...in and Object utilities), and source-code recovery (JavaScript functions store their source text and can be retrieved through toString()).

This section is dedicated to the JavaScript language itself, and not the parts that are specific to Web pages or other host environments. For information about APIs that are specific to Web pages, please see Web APIs and DOM.

The standards for JavaScript are the ECMAScript Language Specification (ECMA-262) and the ECMAScript Internationalization API specification (ECMA-402). As soon as one browser implements a feature, we try to document it. This means that cases where some proposals for new ECMAScript features have already been implemented in browsers, documentation and examples in MDN articles may use some of those new features. Most of the time, this happens between the stages 3 and 4, and is usually before the spec is officially published.

Do not confuse JavaScript with the Java programming language — JavaScript is not "Interpreted Java". Both "Java" and "JavaScript" are trademarks or registered trademarks of Oracle in the U.S. and other countries. However, the two programming languages have very different syntax, semantics, and use.

```
<!DOCTYPE html>
<html>
<head>
  <title>Event Demonstration</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      text-align: center;
    }
    .event-container {
      margin: 20px;
    }
    button {
      padding: 10px 20px;
      font-size: 16px;
      cursor: pointer;
    }
  </style>
</head>
<body>
  <h1>Event Demonstration</h1>

  <div class="event-container">
    <h2>Mouse Events</h2>
    <button id="mouseOverButton">Mouse Over</button>
    <button id="mouseOutButton">Mouse Out</button>
    <p id="mouseEventText">Mouse event will appear here.</p>
  </div>

  <div class="event-container">
    <h2>Keyboard Event</h2>
    <p>Type something in the input field below:</p>
    <input type="text" id="keyboardInput">
    <p id="keyboardEventText">Keyboard event will appear here.</p>
  </div>

  <div class="event-container">
    <h2>Window Events</h2>
    <p>Resize the window to see the event:</p>
    <p id="windowEventText">Window event will appear here.</p>
  </div>

  <div class="event-container">
    <h2>Form Events</h2>
```

```
<form id="demoForm">
  <label for="name">Name:</label>
  <input type="text" id="name" required><br><br>

  <label for="email">Email:</label>
  <input type="email" id="email" required><br><br>

  <button type="submit">Submit</button>
</form>
<p id="formEventText">Form event will appear here.</p>
</div>

<div class="event-container">
  <h2>Additional Events</h2>
  <button id="clickEventButton">Click Me</button>
  <p id="clickEventText">Click event will appear here.</p>

  <input type="text" id="focusEventInput" placeholder="Focus me">
  <p id="focusEventText">Focus event will appear here.</p>
</div>

<script>
const mouseOverButton = document.getElementById("mouseOverButton");
const mouseOutButton = document.getElementById("mouseOutButton");
const mouseEventText = document.getElementById("mouseEventText");

mouseOverButton.addEventListener("mouseover", () => {
  mouseEventText.textContent = "Mouse is over the 'Mouse Over' button!";
});

mouseOutButton.addEventListener("mouseout", () => {
  mouseEventText.textContent = "Mouse is not over any button.";
});
const keyboardInput = document.getElementById("keyboardInput");
const keyboardEventText = document.getElementById("keyboardEventText");
keyboardInput.addEventListener("keydown", (event) => {
  keyboardEventText.textContent = `Key pressed: ${event.key}, Key code:
${event.keyCode}`;
});
const windowEventText = document.getElementById("windowEventText");
window.addEventListener("resize", () => {
  windowEventText.textContent = "Window has been resized!";
});
```

```
const demoForm = document.getElementById("demoForm");
const formEventText = document.getElementById("formEventText");
demoForm.addEventListener("submit", (event) => {
  event.preventDefault(); // Prevent form submission
  formEventText.textContent = "Form submitted!";
});
const clickEventButton = document.getElementById("clickEventButton");
const clickEventText = document.getElementById("clickEventText");

clickEventButton.addEventListener("click", () => {
  clickEventText.textContent = "Button has been clicked!";
});

const focusEventInput = document.getElementById("focusEventInput");
const focusEventText = document.getElementById("focusEventText");

focusEventInput.addEventListener("focus", () => {
  focusEventText.textContent = "Input field is focused!";
});

focusEventInput.addEventListener("blur", () => {
  focusEventText.textContent = "Input field is no longer focused.";
});
</script>
</body>
</html>
```

**Output:**

# Event Demonstration

## Mouse Events

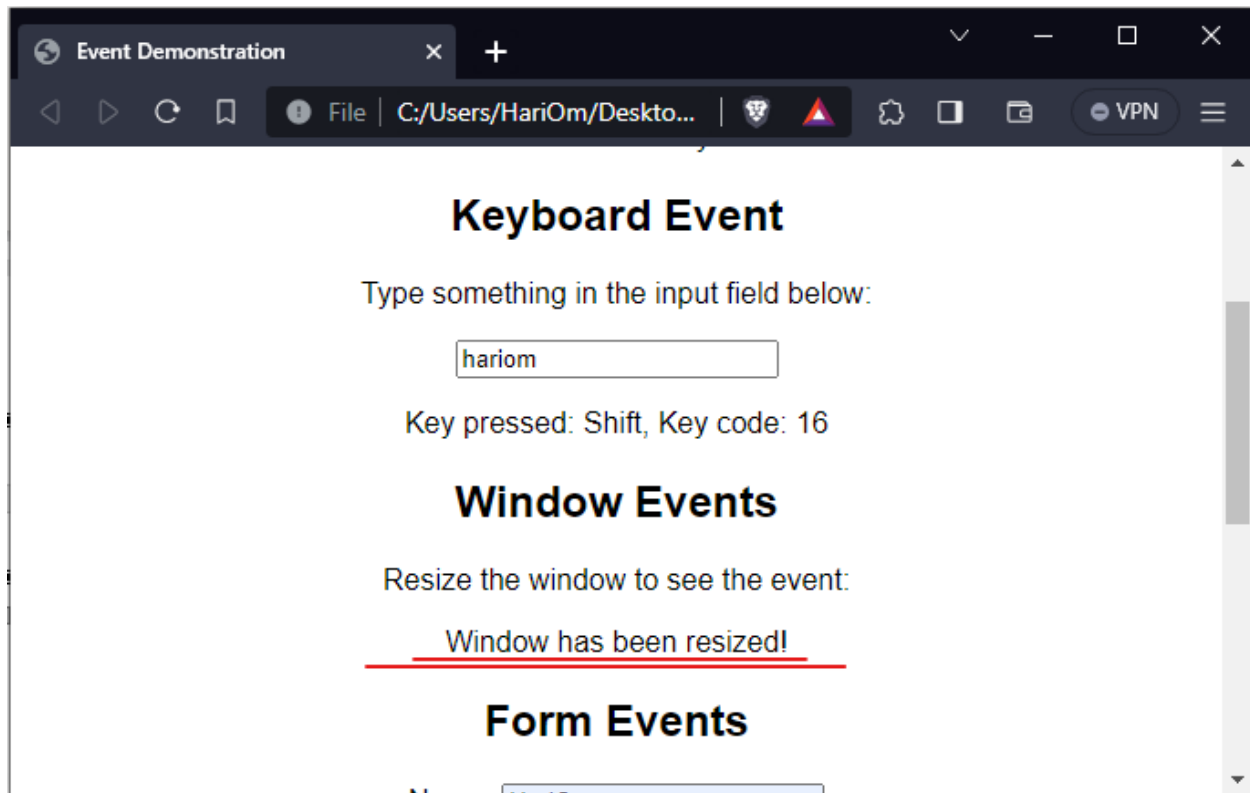
Mouse is over the 'Mouse Over' button!



## Keyboard Event

Type something in the input field below:

Key pressed: Shift, Key code: 16



## Form Events

Name:

Email:

Form submitted!

## Additional Events

Click Me

Button has been clicked!

focus

Input field is focused!

**CONCLUSION:** Through this experiment we learnt events in html & javascript.

**HariOm Shukla****EXPERIMENT – 06****121A3052**

---

**Aim:** To create form & validate using JavaScript.

**THEORY:**

**HTML:**

The Hyper Text Markup Language, or HTML is the standard markup language for documents designed to be displayed in a web browser. It can be assisted by technologies such as Cascading Style Sheets (CSS) and scripting languages such as JavaScript. Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document.

**JavaScript:**

JavaScript (JS) is a lightweight interpreted (or just-in-time compiled) programming language with first-class functions. While it is most well-known as the scripting language for Web pages, many non-browser environments also use it, such as Node.js, Apache CouchDB and Adobe Acrobat. JavaScript is a prototype-based, multi-paradigm, single-threaded, dynamic language, supporting object-oriented, imperative, and declarative (e.g. functional programming) styles.

JavaScript's dynamic capabilities include runtime object construction, variable parameter lists, function variables, dynamic script creation (via eval), object introspection (via for...in and Object utilities), and source-code recovery (JavaScript functions store their source text and can be retrieved through toString()).

This section is dedicated to the JavaScript language itself, and not the parts that are specific to Web pages or other host environments. For information about APIs that are specific to Web pages, please see Web APIs and DOM.

The standards for JavaScript are the ECMAScript Language Specification (ECMA-262) and the ECMAScript Internationalization API specification (ECMA-402). As soon as one browser implements a feature, we try to document it. This means that cases where some proposals for new ECMAScript features have already been implemented in browsers, documentation and examples in MDN articles may use some of those new features. Most of the time, this happens between the stages 3 and 4, and is usually before the spec is officially published.

Do not confuse JavaScript with the Java programming language — JavaScript is not "Interpreted Java". Both "Java" and "JavaScript" are trademarks or registered trademarks of Oracle in the U.S. and other countries. However, the two programming languages have very different syntax, semantics, and use.

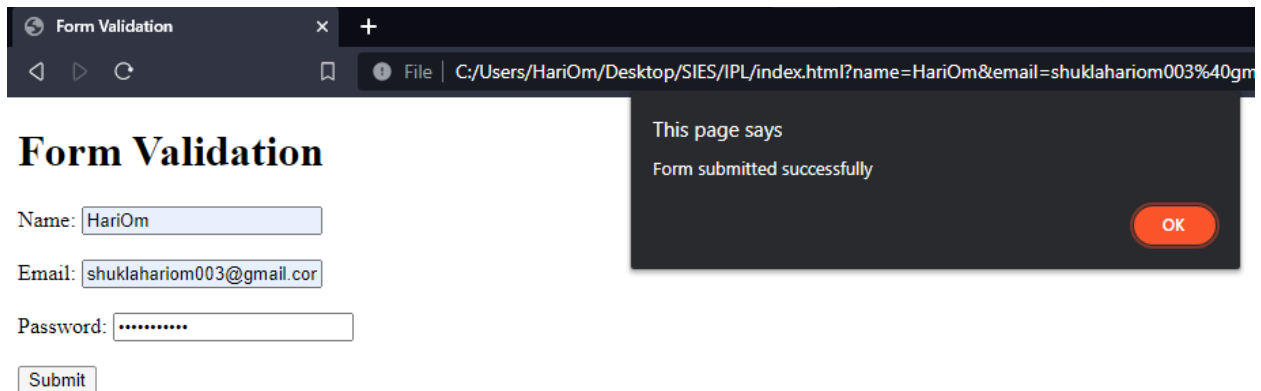
```
<!DOCTYPE html>
<html>
<head>
  <title>Form Validation</title>
</head>
<body>
  <h1>Form Validation</h1>
  <form id="myForm" onsubmit="return validateForm()">
    <label for="name">Name:</label>
    <input type="text" id="name" name="name"><br><br>

    <label for="email">Email:</label>
    <input type="text" id="email" name="email"><br><br>

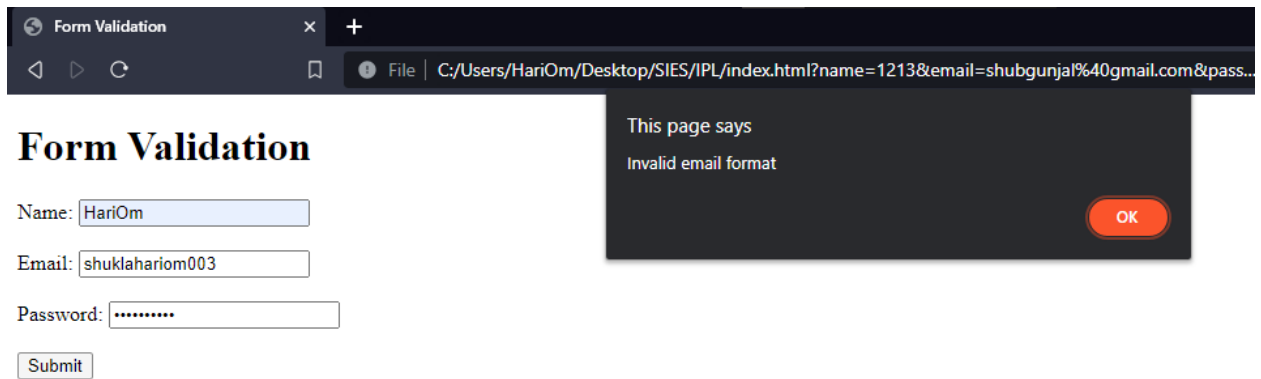
    <label for="password">Password:</label>
    <input type="password" id="password" name="password"><br><br>

    <input type="submit" value="Submit">
  </form>
  <script>
    function validateForm() {
      var name = document.getElementById("name").value;
      var email = document.getElementById("email").value;
      var password = document.getElementById("password").value;

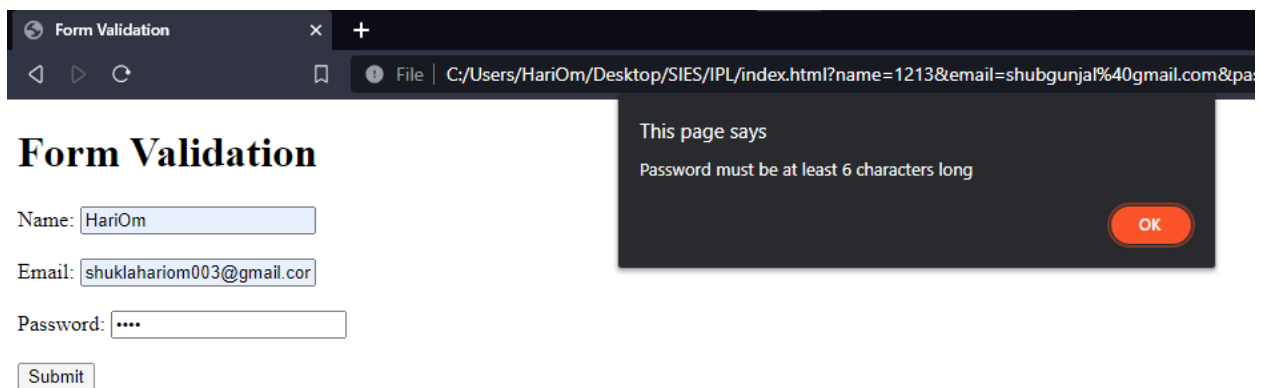
      if (name === "" || email === "" || password === "") {
        alert("All fields must be filled out");
        return false;
      }
      var emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
      if (!emailRegex.test(email)) {
        alert("Invalid email format");
        return false;
      }
      if (password.length < 6) {
        alert("Password must be at least 6 characters long");
        return false;
      }
      alert("Form submitted successfully");
      return true;
    }
  </script>
</body>
</html>
```

**Output:**

The screenshot shows a web browser window titled "Form Validation". The address bar displays the file path: `C:/Users/HariOm/Desktop/SIES/IPL/index.html?name=HariOm&email=shuklahariom003%40gm`. The page content includes a form with the following fields: "Name:" with the value "HariOm", "Email:" with the value "shuklahariom003@gmail.cor", and "Password:" with masked characters "\*\*\*\*\*". A "Submit" button is located below the form. To the right of the form, a dark overlay box contains the text "This page says" followed by "Form submitted successfully" and an "OK" button.



The screenshot shows the same "Form Validation" web browser window. The address bar displays the file path: `C:/Users/HariOm/Desktop/SIES/IPL/index.html?name=1213&email=shubgunjal%40gmail.com&pass...`. The form fields are: "Name:" with "HariOm", "Email:" with "shuklahariom003", and "Password:" with "\*\*\*\*\*". The "Submit" button is present. A dark overlay box on the right displays the message "This page says" followed by "Invalid email format" and an "OK" button.



The screenshot shows the "Form Validation" web browser window. The address bar displays the file path: `C:/Users/HariOm/Desktop/SIES/IPL/index.html?name=1213&email=shubgunjal%40gmail.com&pa`. The form fields are: "Name:" with "HariOm", "Email:" with "shuklahariom003@gmail.cor", and "Password:" with four masked characters "....". The "Submit" button is present. A dark overlay box on the right displays the message "This page says" followed by "Password must be at least 6 characters long" and an "OK" button.

**CONCLUSION:** Through this experiment we learnt to create form & validate it using JavaScript.

**HariOm Shukla****EXPERIMENT – 07****121A3052**

**Aim:** Installation of react & create a class & function component.

**THEORY:**

**REACT**

The React.js framework is an open-source JavaScript framework and library developed by Facebook. It's used for building interactive user interfaces and web applications quickly and efficiently with significantly less code than you would with vanilla JavaScript.

In React, you develop your applications by creating reusable components that you can think of as independent Lego blocks. These components are individual pieces of a final interface, which, when assembled, form the application's entire user interface.

React's primary role in an application is to handle the view layer of that application just like the V in a model-view-controller (MVC) pattern by providing the best and most efficient rendering execution. Rather than dealing with the whole user interface as a single unit, React.js encourages developers to separate these complex UIs into individual reusable components that form the building blocks of the whole UI. In doing so, the ReactJS framework combines the speed and efficiency of JavaScript with a more efficient method of manipulating the DOM to render web pages faster and create highly dynamic and responsive web applications.

React has two types of components: **functional (stateless)** and **class (stateful)**. Functional components are JavaScript functions that use React hooks to provide the equivalent functionality as class components. They are constructed with single props object arguments and return React elements. Function components were the only way to track state and lifecycle on a React component before React 16.8, but with the addition of Hooks, they are almost equivalent to class components. Functional component names must begin with an uppercase letter, while a JavaScript function name can start with a lowercase letter. Hooks were not introduced into React until version 16.8, but they have effectively made class components redundant.

**Code:**

**App.js:**

```
import './App.css';
import Function from './Component/Function';
import Class1 from './Component/Class1';
import Hook2useState from './Components/Hook2useState';

function App(){
  return(
    <div className="App">
      <>
```

```
    <Function/>
    <Class1/>
  </>
  <header className="App-header">
    <h1>HariOm Shukla </h1>
    <a
      className="App-link"
      href="https://react.org"
      target="_blank"
      rel="noopener noreferrer"
    >
      Learn React
    </a>
  </header>
</div>
);
}
export default App;
```

**App.css:**

```
.App{
  text-align: left;
  background-color: #282c34;
  color: white;
  font-size: 20px;
  padding-top: 200px;
}

.App-header{
  min-height: 100vh;
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content:center;
  font-size: calc(10px + 2vmin);
  color: white;
}

.app-link{
  color: #61dafb;
}
```

**Class1.js:**

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

class Class1 extends Component{
  WebGL2RenderingContext(){
    return(
      <div>
        <p><h1>This is class Component</h1></p><br></br>
        <ul>
          <li>X</li>
          <li>Y</li>
          <li>Z</li>
        </ul>
      </div>
    );
  }
}

export default Class1;
```

**Function.js:**

```
import React from "react";

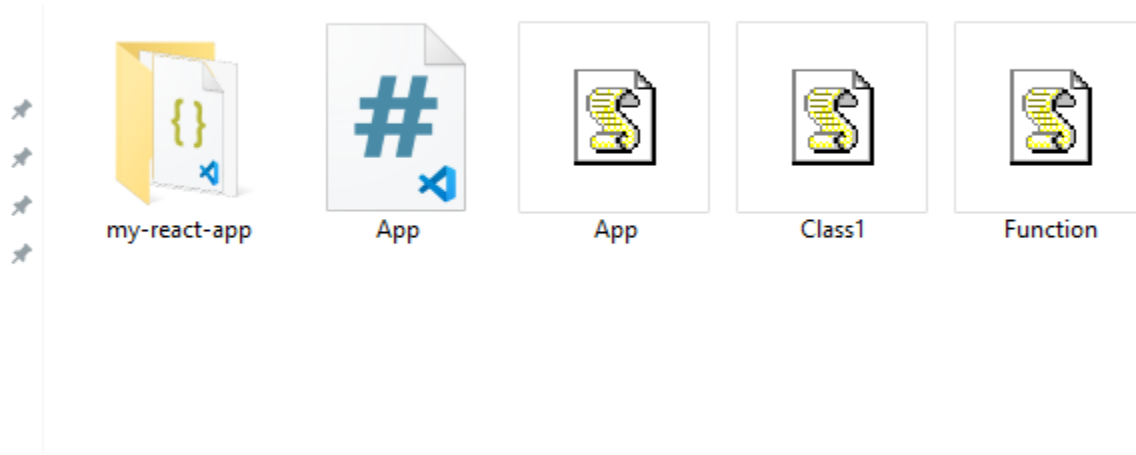
function Function(){
  return(
    <div>
      <p><h1>This is function component.</h1></p><br></br>
      <ul>
        <li>A</li>
        <li>B</li>
        <li>C</li>
      </ul>
    </div>
  );
}

export default Function;
```



**Output:**

HariOm



```
0% npm install react react-dom react-scripts cra-template
Microsoft Windows [Version 10.0.19045.3324]
(c) Microsoft Corporation. All rights reserved.

C:\Users\exam\Desktop\HariOm>npm -v
9.6.7

C:\Users\exam\Desktop\HariOm>node -v
v18.17.1

C:\Users\exam\Desktop\HariOm>npx create-react-app my-react-app

Creating a new React app in C:\Users\exam\Desktop\HariOm\my-react-app.

Installing packages. This might take a couple of minutes.
Installing react, react-dom, and react-scripts with cra-template...

[██████████] - reify:fsevents: sill reify mark deleted [
```

C:\Windows\System32\cmd.exe

```
at Script.runInThisContext (node:vm:123:12)
at Object.runInThisContext (node:vm:299:38)
at node:internal/process/execution:79:19
at [eval]-wrapper:6:22 {
  status: 1,
  signal: null,
  output: [ null, null, null ],
  pid: 8012,
  stdout: null,
  stderr: null
}
```

Installing template dependencies using npm...

added 69 packages, and changed 1 package in 25s

245 packages are looking for funding

run `npm fund` for details

Removing template package using npm...

removed 1 package, and audited 1522 packages in 4s

245 packages are looking for funding

run `npm fund` for details

6 high severity vulnerabilities

To address all issues (including breaking changes), run:  
npm audit fix --force

Run `npm audit` for details.

Success! Created my-react-app at C:\Users\exam\Desktop\HariOm\my-react-app  
Inside that directory, you can run several commands:

**npm start**

Starts the development server.

**npm run build**

Bundles the app into static files for production.

**npm test**

Starts the test runner.

**npm run eject**

Removes this tool and copies build dependencies, configuration files  
and scripts into the app directory. If you do this, you can't go back!

We suggest that you begin by typing:

**cd my-react-app**

**npm start**

Happy hacking!

C:\Users\exam\Desktop\HariOm>\_

```
C:\Users\exam\Desktop\HariOm>cd my-react-app
C:\Users\exam\Desktop\HariOm\my-react-app>npm start

> my-react-app@0.1.0 start
> react-scripts start

(node:3120) [DEP_WEBPACK_DEV_SERVER_ON_AFTER_SETUP_MIDDLEWARE] DeprecationWarning: 'onAfterSetupMiddleware' option is deprecated. Please use the 'setupMiddlewares' option.
(Use 'node --trace-deprecation ...' to show where the warning was created)
(node:3120) [DEP_WEBPACK_DEV_SERVER_ON_BEFORE_SETUP_MIDDLEWARE] DeprecationWarning: 'onBeforeSetupMiddleware' option is deprecated. Please use the 'setupMiddlewares' option.
Starting the development server...

One of your dependencies, babel-preset-react-app, is importing the
"@babel/plugin-proposal-private-property-in-object" package without
declaring it in its dependencies. This is currently working because
"@babel/plugin-proposal-private-property-in-object" is already in your
node_modules folder for unrelated reasons, but it may break at any time.

babel-preset-react-app is part of the create-react-app project, which
is not maintained anymore. It is thus unlikely that this bug will
ever be fixed. Add "@babel/plugin-proposal-private-property-in-object" to
your devDependencies to work around this error. This will make this message
go away.
Compiled successfully!

You can now view my-react-app in the browser.

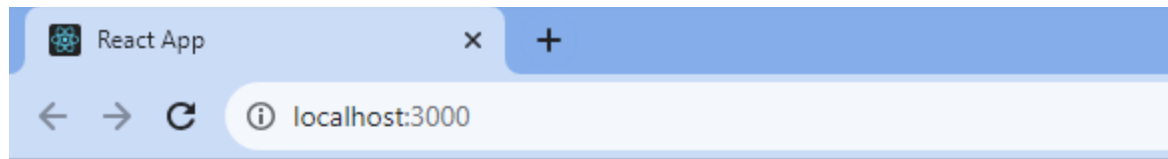
  Local:            http://localhost:3000
  On Your Network:  http://10.0.2.176:3000

Note that the development build is not optimized.
To create a production build, use npm run build.

webpack compiled successfully
```

```
EXPLORER  ...  JS App.js  X
HARIOM
  my-react-app
    node_modules
    public
    src
      App.css
      App.js
      App.test.js
      index.css
      index.js
      logo.svg
      reportWebVitals.js
      setupTests.js
      .gitignore
      package-lock.json
      package.json
      README.md
      App.css
      App.js
      Class1.js
      Function.js

my-react-app > src > JS App.js > ...
1  import React from 'react';
2  import './App.css';
3
4  function App() {
5    return (
6      <h1> Hello World! </h1>
7    );
8  }
9
10 export default App;
11
```



# Hello World!

**CONCLUSION:** Installation of react and to create a class & function component was done successfully.

**HariOm Shukla****Experiment No: 8****121A3052**

---

**Aim: WAP to handle click events by building a nav menu using router.**

**Theory:**

### **1. ReactJS:**

React (also known as React.js or ReactJS) is a free and open-source front-end JavaScript library for building user interfaces or UI components. It is maintained by Facebook and a community of individual developers and companies. React can be used as a base in the development of single-page or mobile applications. However, React is only concerned with state management and rendering that state to the DOM, so creating React applications usually requires the use of additional libraries for routing, as well as certain client-side functionality.

### **2. React Router:**

At its core, what React Router does is conditionally render certain components to display depending on the route being used in the URL (/ for the home page, /about for the about page, etc.). To install react router we need to type 'npm install react-router-dom' in the command line and it will install the react router for us. It has various components like BrowserRouter, Route & Switch. BrowserRouter is the tag where we write everything that we want to be rendered. Now comes the Switch component, this component is used to ensure that only one component is rendered at a time. Lastly the Route tag, this tag is used to simply load the component.

### **What is < Link> component?**

This component is used to create links which allow to navigate on different URLs and render its content without reloading the webpage.

## <Link> vs <NavLink>

The Link component allows navigating the different routes on the websites, whereas NavLink component is used to add styles to the active routes.

## Benefits Of React Router

The benefits of React Router is given below:

In this, it is not necessary to set the browser history manually.

Link uses to navigate the internal links in the application. It is similar to the anchor tag.

It uses Switch feature for rendering.

The Router needs only a Single Child element.

In this, every component is specified in .

## CODE:

```
JS App.js  X  JS Counter.js  JS Home.js  JS About.js  JS Users.js
myapp > src > JS App.js > ...
1  import React from 'react';
2  import { Component } from 'react';
3  import { BrowserRouter as Router, Routes, Route } from 'react-router-dom';
4  import Counter from './components/Counter';
5  import About from './components/About';
6  import Users from './components/Users';
7  import Home from './components/Home';
8  class App extends Component {
9    render() {
10     return (
11       <>
12       <Router>
13       <Routes>
14       <Route path="/" element={<Home />} />
15       <Route path="/about" element={<About />} />
16       <Route path="/users" element={<Users />} />
17       <Route path="/Counter" element={<Counter />} />
18       </Routes>
19       </Router>
20     </>
21   );
22   }
23 }
24 export default App;
25
26
```

```
JS App.js JS Counter.js X JS Home.js JS About.js JS Users.js
myapp > src > components > JS Counter.js > Counter
1 import React from "react";
2 function Counter()
3 {
4   return(
5     <div>
6       <h1>This is Counter page</h1>
7     </div>
8   );
9 }
10 export default Counter;
```

```
JS App.js JS Counter.js JS Home.js X JS About.js JS Users.js
myapp > src > components > JS Home.js > default
1 import React from "react";
2 import { Link } from 'react-router-dom';
3 function Home () {
4
5   return <div><nav>
6     <ul>
7       <li> <Link to="/">Home</Link></li>
8       <li> <Link to="/about">About</Link></li>
9       <li> <Link to="/users">Users</Link></li>
10      <li> <Link to="/Counter">Count</Link></li>
11    </ul>
12    </nav>
13  </div>
14 }
15 export default Home;
```

```
JS App.js JS Counter.js JS Home.js JS About.js X JS Users.js
myapp > src > components > JS About.js > ...
1  import React from "react";
2  function About()
3  {
4      return(
5          <div>
6              <h1>This is About page</h1>
7          </div>
8      );
9  }
10 export default About;
11
12
```

```
JS App.js JS Counter.js JS Home.js JS About.js JS Users.js X
myapp > src > components > JS Users.js > Users
1  import React from "react";
2  function Users()
3  {
4      return(
5          <div>
6              <h1>This is Users page</h1>
7          </div>
8      );
9  }
10 export default Users;
```

- [Home](#)
- [About](#)
- [Users](#)
- [Count](#)



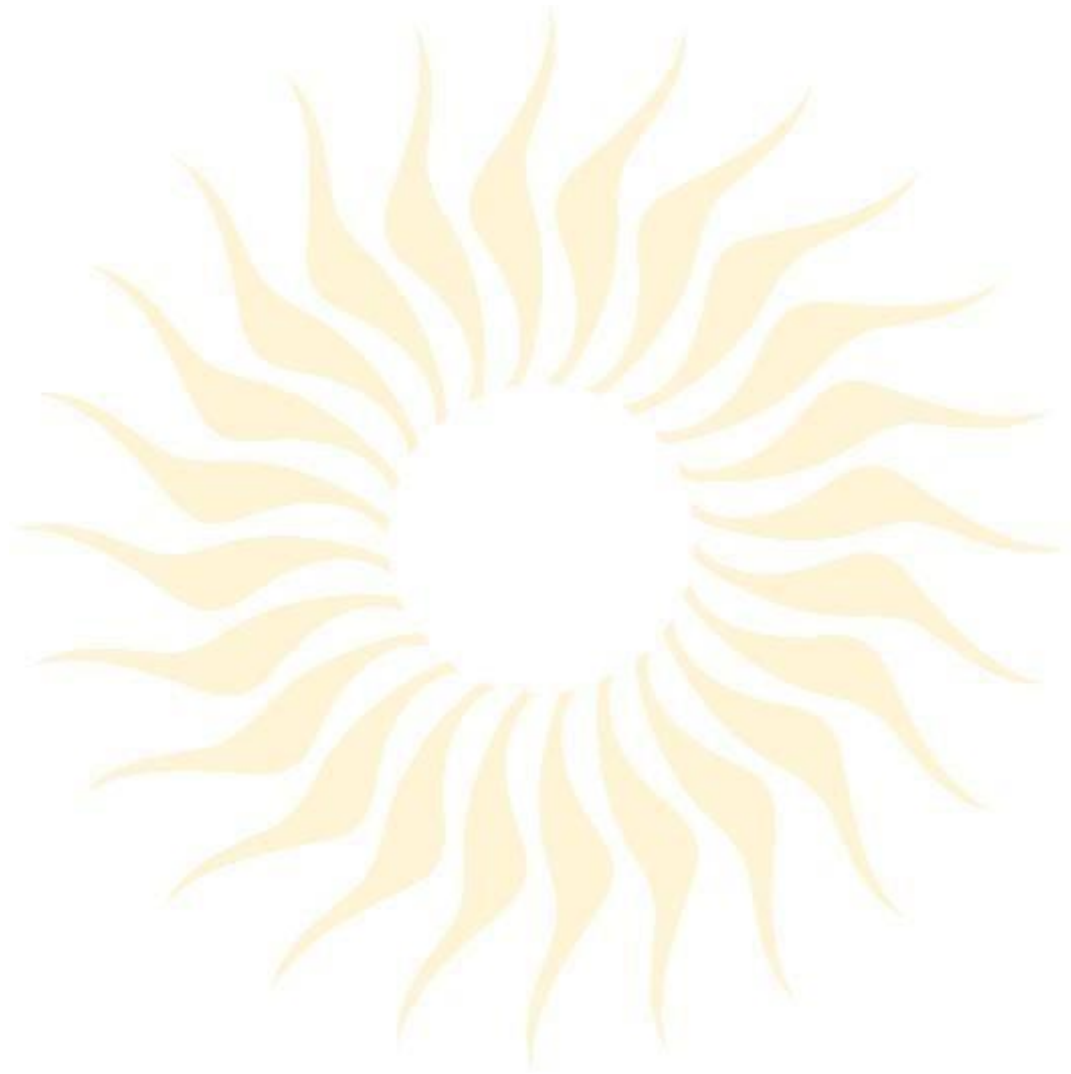
**OUTPUT:**

**This is Users page**

**This is About page**

**This is Counter page**

**CONCLUSION:** We have successfully WAP to handle click events by building a nav Menu using Router.



**HariOm Shukla****Experiment No – 9****121A3052**

**AIM:** Write a program to implement Counter program using Class and hook in React JS.

**THEORY:**

1. **ReactJS:**

React (also known as React.js or ReactJS) is a free and open-source front-end JavaScript library for building user interfaces or UI components. It is maintained by Facebook and a community of individual developers and companies. React can be used as a base in the development of single-page or mobile applications. However, React is only concerned with state management and rendering that state to the DOM, so creating React applications usually requires the use of additional libraries for routing, as well as certain client-side functionality.

2. **Class-Based components:**

React **class-based components** are the bread and butter of most modern web apps built in ReactJS. These components are simple classes (made up of multiple functions that add functionality to the application). All **class-based components** are child classes for the Component class of ReactJS. The main feature of **class-based components** that distinguish them from functional components is that they have access to a **state** which dictates the current behaviour and appearance of the component. This **state** can be modified by calling the **setState() function**. One or more variables, arrays, or objects defined as part of the state can be modified at a time with the **setState() function**.

3. **Hooks:**

React doesn't offer a way to "attach" reusable behaviour to a component. With Hooks, you can extract stateful logic from a component so it can be tested independently and reused. **Hooks allow you to reuse stateful logic without changing your component hierarchy.** This makes it easy to share Hooks among many components. The **useState()** is a Hook that allows you to have state variables in functional components. The **useState** hook is a special function that takes the initial state as an argument and returns an array of two entries.

**CODE:**

**App.js**

```
import './App.css';
import Function from './Components/Function';
import Class1 from './Components/Class1';
import Hook2useState from './Components/Hook2useState';
import Counter from './Component/Counter';
```

```
function App(){
  return(
    <div className ="App">
      <>
        <Hook2useState/>
      </>
      <header className="App-header">
        </header>
      </div>
    );
}
export default App;
```

### **Hook2useState.js**

```
import React,{useState} from "react";
const Hook2useState=()=>{
  const [counter,setCounter]=useState(0);
  const increment=()=>{
    setCounter(counter+1)
  };
  return(
    <div>
      {counter}<br/>
      <button onClick={increment}>Increment</button>
    </div>
  );
}
export default Hook2useState;
```

### **Counter.js**

```
import React from 'react';

class Counter extends React.Component{
  constructor(){
    super();
    this.state={
      count:0,
    }
    this.increment= this.increment.bind(this);
    this.decrement= this.decrement.bind(this);
  }

  increment(){
    this.setState(previousValue=>({
      count: previousValue.count+1,
    }));
  }

  decrement(){
```

```
        this.setState(previousValue=>({
            count: previousValue.count-1,
        }));
    }

    render(){
        return(
            <div className="counter">
                <h1>{this.state.count}</h1>
                <button onClick={this.increment}>+</button>
                <button onClick={this.decrement}>-</button>
            </div>
        );
    }
}

export default Counter;
```

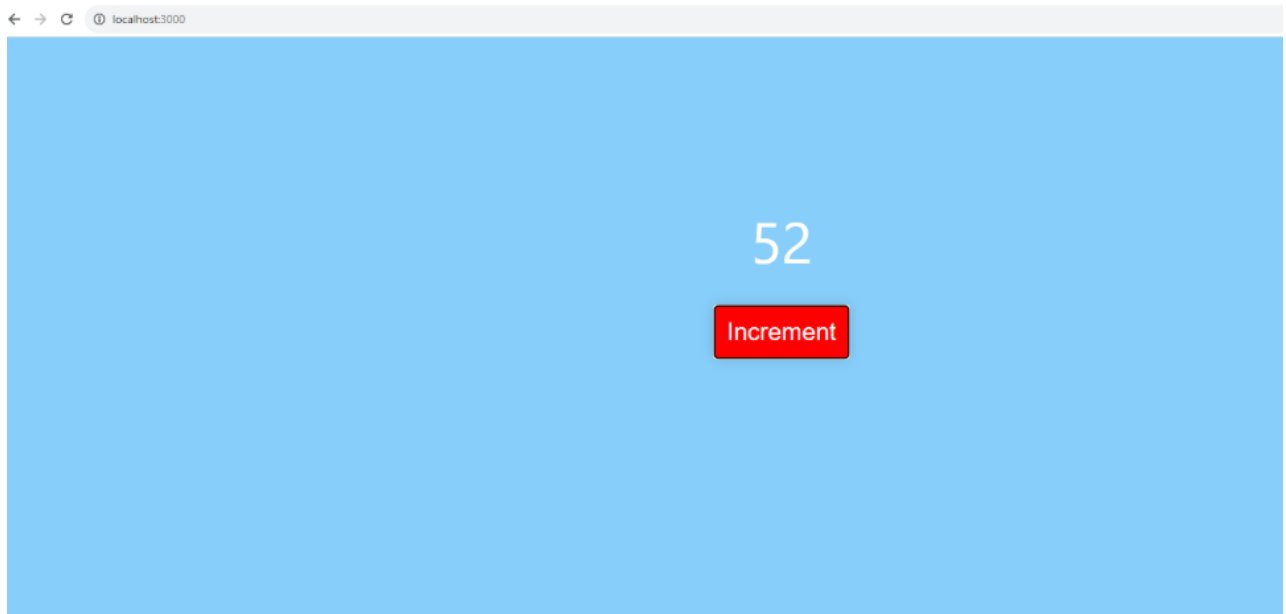
**App.css**

```
.App{
    text-align: left;
    background-color:#282c34 ;
    color: white;
    font-size: 20px;
    padding-top: 200px;
    padding-left: 100px;
}

.App-header{
    min-height: 100vh;
    display: flex;
    flex-direction: column;
    align-items: center;
    justify-content: center;
    font-size: calc(10px + 2vmin);
    color: white;
}

.App-link{
    color: #61dafb;
}

}
```

**OUTPUT:****CONCLUSION:**

We have successfully implemented Counter program using Class and hook in React JS.

**HariOm Shukla****EXPERIMENT – 10****121A3052**

**AIM:** Write a program to implement file system in Node.js

**THEORY:**

**What is Node.js?**

- Node.js is an open-source server environment.
- Node.js is free.
- Node.js runs on various platforms (Windows, Linux, Unix, Mac OS X, etc.)
- Node.js uses JavaScript on the server.

Node.js is an open-source, cross-platform, back-end JavaScript runtime environment that runs on a JavaScript Engine (i.e., V8 engine) and executes JavaScript code outside a web browser, which was designed to build scalable network applications.

Node.js lets developers use JavaScript to write command line tools and for server-side scripting—running scripts server-side to produce dynamic web page content before the page is sent to the user's web browser. Consequently, Node.js represents a "JavaScript everywhere" paradigm,[6] unifying webapplication development around a single programming language, rather than different languages for server-side and client-side scripts.

Node.js has an event-driven architecture capable of asynchronous I/O. These design choices aim to optimize throughput and scalability in web applications with many input/output operations, as well as for real-time Web applications (e.g., real-time communication programs and browser games).

The Node.js distributed development project was previously governed by the Node.js Foundation and has now merged with the JS Foundation to form the OpenJS Foundation, which is facilitated by the Linux Foundation's Collaborative Projects program.

**What is a Module in Node.js?**

Consider modules to be the same as JavaScript libraries. A set of functions you want to include in your application.



<https://upload.wikimedia.org/wikipedia>

Node.js was written initially by Ryan Dahl in 2009,[24] about thirteen years after the introduction of the first server-side JavaScript environment, Netscape's LiveWire Pro Web. The initial release supported only Linux and Mac OS X. Its development and maintenance were led by Dahl and later sponsored by Joyent.

Dahl criticized the limited possibilities of the most popular web server in 2009, Apache HTTP Server, to handle a lot of concurrent connections (up to 10,000 and more) and the most common way of creating code (sequential programming), when code either blocked the entire process or implied multiple execution stacks in the case of simultaneous connections.

In June 2011, Microsoft and Joyent implemented a native Windows version of Node.js. The first Node.js build supporting Windows was released in July 2011.

In January 2012, Dahl stepped aside, promoting coworker and npm creator Isaac Schlueter to manage the project. In January 2014, Schlueter announced that Timothy J. Fontaine would lead the project.



In September 2022, Node.js 18.9.0 was released.[44] Built- in Modules

Node.js has a set of built-in modules which you can use without any further installation.

Look at our Built-in Modules Reference for a complete list of modules Common

use for the File System module:

- Read files
- Create files
- Update files
- Delete files
- Rename files

### **Read Files**

The `fs.readFile()` method is used to read files on your computer.

### **Create Files**

The File System module has methods for creating new files

`fs.appendFile()` `fs.open()` `fs.writeFile()`

The `fs.appendFile()` method appends specified content to a file. If the file does not exist, the file will be created:

### **Update Files**

The File System module has methods for updating files

`fs.appendFile()` `fs.writeFile()`

The `fs.appendFile()` method appends the specified content at the end of the specified file.

### **Delete Files**

To delete a file with the File System module, use the `fs.unlink()` method. The

`fs.unlink()` method deletes the specified file.

## CODE:

```
const fs = require("fs"); const path = require("path")

const dirPath = path.join(__dirname, 'test'); const
filePath = dirPath + "\\node.txt";

fs.writeFileSync(filePath, `${filePath} Created !!`);

const data = fs.readFileSync(filePath, 'utf-8'); console.log(data);

const apErr = fs.appendFileSync(filePath, "\nThis is append data")
if (apErr) console.log(apErr); else console.log("Appended");

const renameErr = fs.renameSync(filePath, dirPath + "\\newtest.txt");
if (renameErr) console.log(renameErr); else console.log("Renamed");

const delErr = fs.unlinkSync(dirPath + "\\newtest.txt");
if (delErr) console.log(delErr); else
console.log("deleted");
```

## OUTPUT:

```
PS C:\node_experiments> node node.js
This is sample text file created
PS C:\node_experiments> node node.js
This is sample text file created
sample.txt File is updated
PS C:\node_experiments> node node.js
filename is updated
sample.txt File is updated
This is sample text file created
PS C:\node_experiments> node node.js
sample.txt File is updated
This is sample text file created
```

**CONCLUSION:** The implementation of the file system in Node.js was performed successfully.

NAME: Hariom Shukla.

PRN: 121A3052

CLASS: TE IT

BATCH: E3

## EXPERIMENT 11

**Aim:** Write a program to implement and understand the 'file server' in Nodejs.

### Theory:

A basic necessity for most http servers is to be able to serve static files. Thankfully, it is not that hard to do in Node.js. First you read the file, then you serve the file.

### NodeJS:

Node.js is an open-source, cross-platform, back-end JavaScript runtime environment that runs on the V8 engine and executes JavaScript code outside a web browser. Node.js lets developers use JavaScript to write command line tools and for server-side scripting—running scripts server-side to produce dynamic web page content before the page is sent to the user's web browser. Consequently, Node.js represents a "JavaScript everywhere" paradigm,[6] unifying web-application development around a single programming language, rather than different languages for server-side and client-side scripts. Node.js was written initially by Ryan Dahl in 2009,[25] about thirteen years after the introduction of the first server-side JavaScript environment, Netscape's LiveWire Pro Web. The initial release supported only Linux and Mac OS X. Its development and maintenance was led by Dahl and later sponsored by Joyent. Node.js allows the creation of Web servers and networking tools using JavaScript and a collection of "modules" that handle various core functionalities. Modules are provided for file system I/O, networking (DNS, HTTP, TCP, TLS/SSL, or UDP), binary data (buffers), cryptography functions, data streams, and other core functions. Node.js's modules use an API designed to reduce the complexity of writing server applications.

### 'http' Module:

To make HTTP requests in Node.js, there is a built-in module HTTP in Node.js to transfer data over the HTTP. To use the HTTP server in node, we need to require the HTTP module. The HTTP module creates an HTTP server that listens to server ports and gives a response back to the client.

Syntax: `var http = require('http');` We can create a HTTP server with the help of `http.createServer()` method.

### 'fs' File System Module:

To handle file operations like creating, reading, deleting, etc., Node.js provides an inbuilt module called FS (File System). Node.js gives the functionality of file I/O by providing wrappers around the standard POSIX functions. All file system operations can have synchronous and asynchronous forms depending upon user requirements. To use this File System module, use the `require()` method: `var fs = require('fs');`

**Code:**

```
const http=require('http'); const fs=require('fs');
const port=3000 const
server=http.createServer(function(req,res){
res.writeHead(200,{ 'content-type':'text/html'})
  fs.readFile('page.html',function(error,data){
if(error){

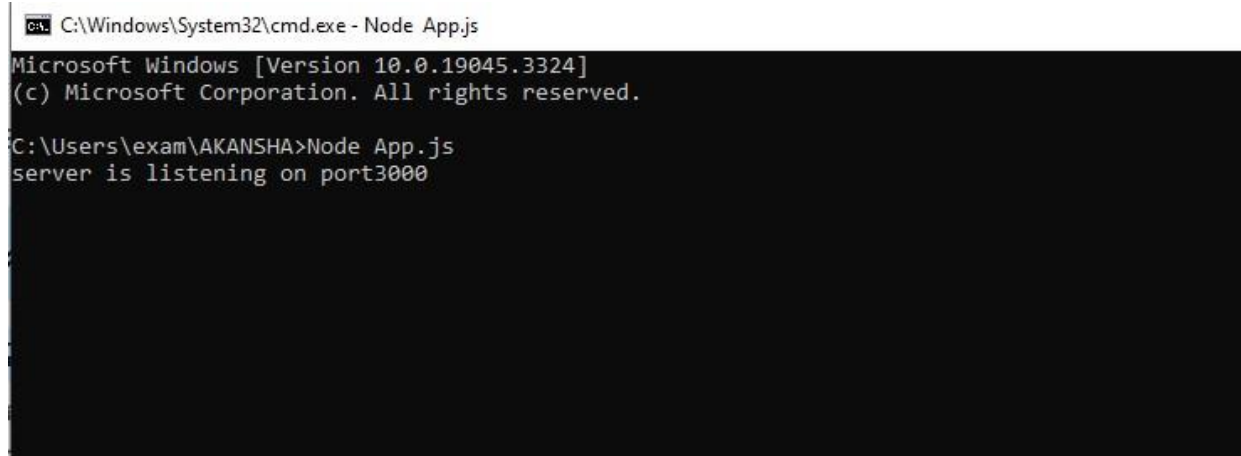
    res.write(404)
    res.write('error: File not Found')
  }else{
res.write(data)
  }
res.end()
  }) })
server.listen(port,function(error){
  if(error){
    console.log('something went wrong',error)
  }
else{
    console.log('server is listening on port'+port)
  }
})
```

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <p>          my name is
akansha sukale    roll no. is
120A3057    from SIES
  THANK YOU!!!!!!!!!!!!!!
  </p>
</body>
</html>
```

```
const http=require('http'); const fs=require('fs');
const port=3000 const
server=http.createServer(function(req,res){
res.writeHead(200,{ 'content-type':'text/html'})
  fs.readFile('page.html',function(error,data){
if(error){      res.write(404)
    res.write('error: File not Found')
```

```
    }else{  
res.write(data)  
    }  
res.end()  
  })  
})  
  
server.listen(port,function(error){  
  if(error){  
    console.log('something went wrong',error)  
  }  
  else{  
    console.log('server is listening on port'+port)  
  }  
})
```

### Output:



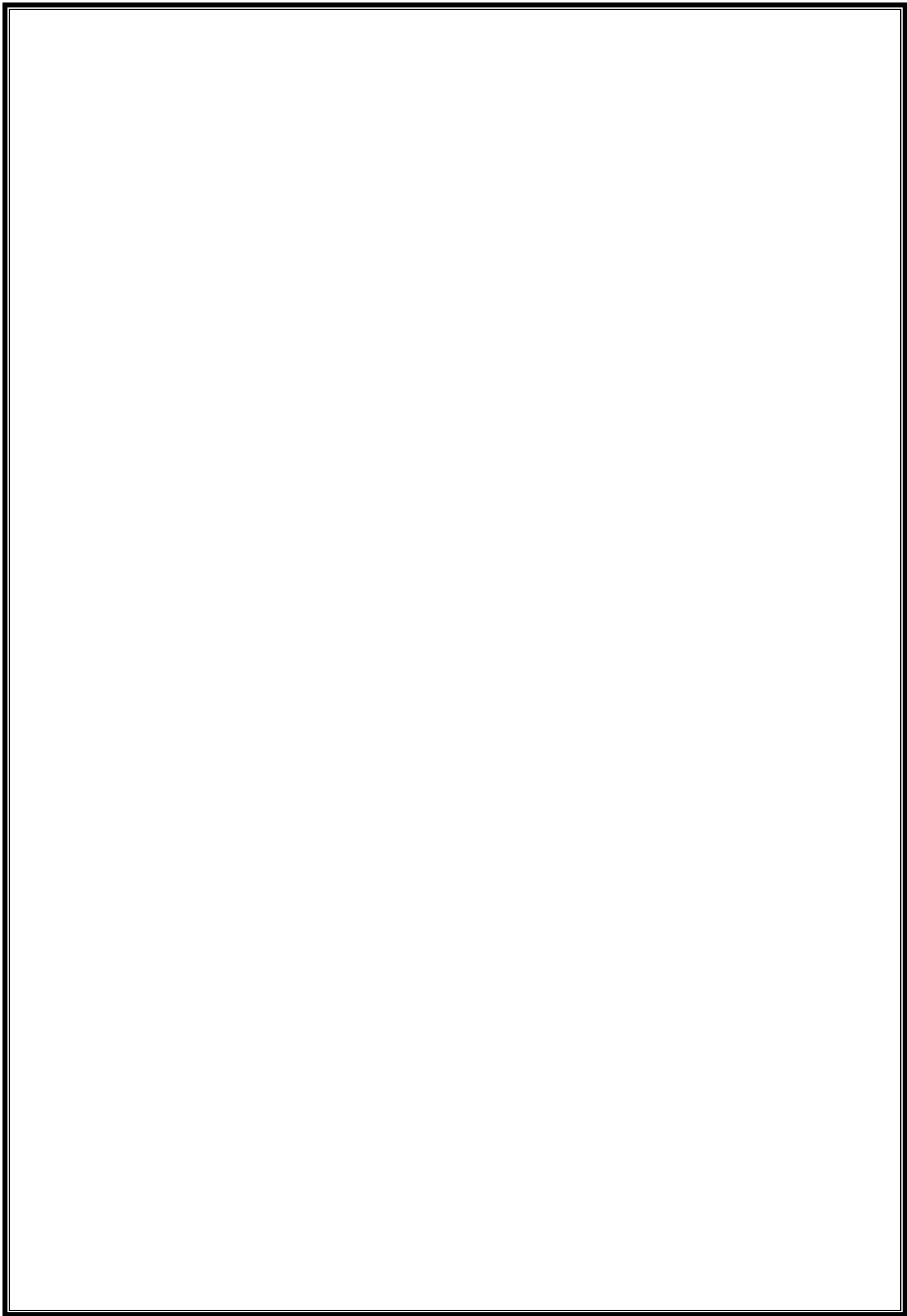
```
C:\Windows\System32\cmd.exe - Node App.js  
Microsoft Windows [Version 10.0.19045.3324]  
(c) Microsoft Corporation. All rights reserved.  
  
C:\Users\exam\AKANSHA>Node App.js  
server is listening on port3000
```



← → ↻ ⓘ localhost:3000

my name is akansha sukale roll no. is 120A3057 from SIES THANK YOU!!!!!!!!!!!!!!

**Conclusion:** We have successfully implemented and understood the ‘file server’ in Nodejs.



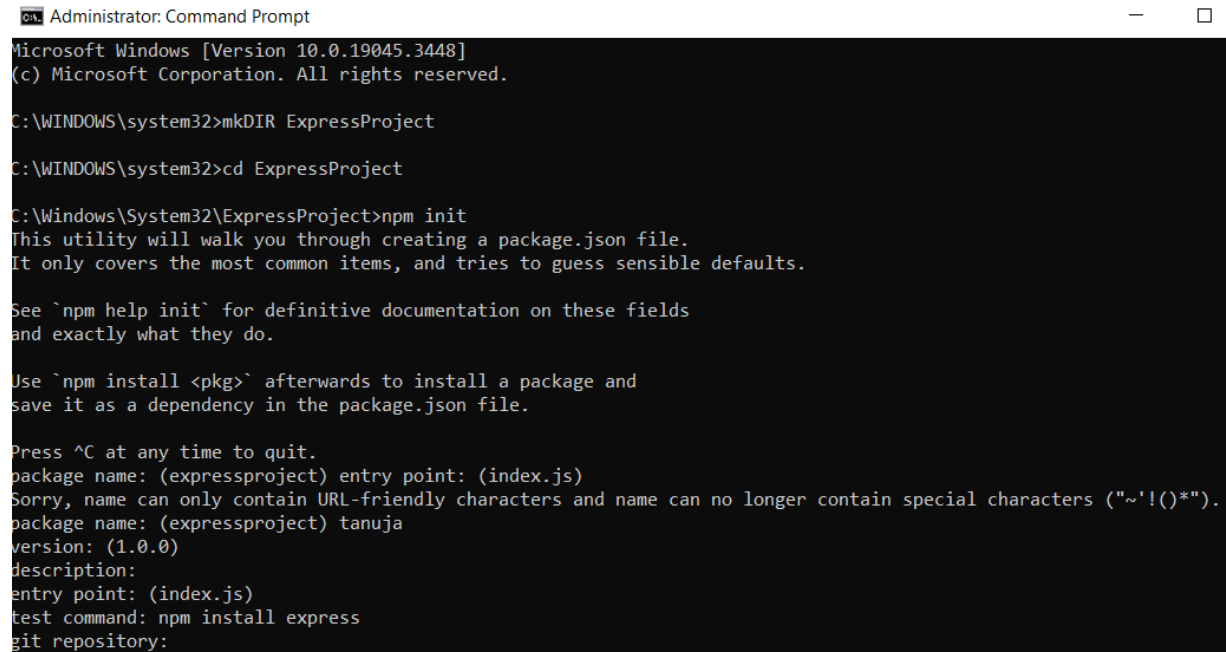
**HariOm Shukla****Experiment No. 12****121A3052**

---

**Aim:** Install Express JS Framework and Write a Program to print Hello World!

**THEORY:** Express JS: Express is a fast, assertive, essential and moderate web framework of Node.js. You can assume express as a layer built on the top of the Node.js that helps manage a server and routes. It provides a robust set of features to develop web and mobile applications. Some core features of Express framework: It can be used to design single-page, multi-page and hybrid web applications. It allows to setup middle wares to respond to HTTP Requests. It defines a routing table which is used to perform different actions based on HTTP method and URL. It allows to dynamically render HTML Pages based on passing arguments to templates. Installation process of Express JS: Open a new terminal in VS Code In the command line type the following command: `npm install express` Hence express JS framework is installed in your respected systems

## INSTALLATION OF EXPRESS:



Administrator: Command Prompt

Microsoft Windows [Version 10.0.19045.3448]  
(c) Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>mkdir ExpressProject

C:\WINDOWS\system32>cd ExpressProject

C:\Windows\System32\ExpressProject>npm init

This utility will walk you through creating a package.json file.  
It only covers the most common items, and tries to guess sensible defaults.

See `npm help init` for definitive documentation on these fields  
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and  
save it as a dependency in the package.json file.

Press ^C at any time to quit.

package name: (expressproject) entry point: (index.js)

Sorry, name can only contain URL-friendly characters and name can no longer contain special characters ("~!()\*").

package name: (expressproject) tanuja

version: (1.0.0)

description:

entry point: (index.js)

test command: npm install express

git repository:



Administrator: Command Prompt

```
test command: npm install express
git repository:
keywords:
author:
license: (ISC)
About to write to C:\Windows\System32\ExpressProject\package.json:
```

```
{
  "name": "tanuja",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "npm install express"
  },
  "author": "",
  "license": "ISC"
}
```

Is this OK? (yes) yes

```
C:\Windows\System32\ExpressProject>npm install express
```

```
added 58 packages, and audited 59 packages in 6s
```

```
8 packages are looking for funding
  run `npm fund` for details
```

```
8 packages are looking for funding
  run `npm fund` for details
```

```
found 0 vulnerabilities
```

```
C:\Windows\System32\ExpressProject>node get_example1.js
```

```
node:internal/modules/cjs/loader:1080
  throw err;
  ^
```

```
Error: Cannot find module 'C:\Windows\System32\ExpressProject\get_example1.js'
    at Module._resolveFilename (node:internal/modules/cjs/loader:1077:15)
    at Module._load (node:internal/modules/cjs/loader:922:27)
    at Function.executeUserEntryPoint [as runMain] (node:internal/modules/run_main:81:12)
    at node:internal/main/run_main_module:23:47 {
  code: 'MODULE_NOT_FOUND',
  requireStack: []
}
```

```
Node.js v18.17.1
```

**CODE:**

```
const express = require('express');
const app = express();
const port = 3000; // Set the port for the server

// Define a route to handle the root URL
app.get('/', (req, res) => {
  res.send('Hello, World!');
});

// Start the server and listen on the specified port
app.listen(port, () => {
  console.log(`Server is running at http://localhost:${port}`);
});
```

**OUTPUT:**

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

PS C:\Users\exam\Desktop\121A3062> node exp12.js

node:internal/modules/cjs/loader:1080

throw err;

^

Error: Cannot find module 'express'

Require stack:

- C:\Users\exam\Desktop\121A3062\exp12.js

at Module.\_resolveFilename (node:internal/modules/cjs/loader:1077:15)

at Module.\_load (node:internal/modules/cjs/loader:922:27)

at Module.require (node:internal/modules/cjs/loader:1143:19)

at Object.<anonymous> (C:\Users\exam\Desktop\121A3062\exp12.js:1:17)

at Module.\_compile (node:internal/modules/cjs/loader:1256:14)

```
  requireStack: [ 'C:\\Users\\exam\\Desktop\\121A3062\\exp12.js' ]  
}
```

```
Node.js v18.17.1
```

```
PS C:\Users\exam\Desktop\121A3062> npm i express
```

```
added 58 packages in 2s
```

```
8 packages are looking for funding
```

```
  run `npm fund` for details
```

```
PS C:\Users\exam\Desktop\121A3062> node exp12.js
```

```
Server is running at http://localhost:3000
```

```
█
```

---

Hello, World!

**Conclusion:** Hence we have successfully installed express.