NODE JS PRACTICAL

Practical 1

AIM: Write a Program to pass a message "Hello Node JS" using Node JS

Line of Code:

```
console.log("Hello Node JS");
```

Output Screen:

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS                                   powershell

Warning: PowerShell detected that you might be using a screen reader and has disabled PSReadLine for compatibility purposes. If you
want to re-enable it, run 'Import-Module PSReadLine'.

PS C:\Program Files\Sanyukta> Node First.js
Hello Node JS
PS C:\Program Files\Sanyukta>
```

Practical 2

AIM: Write a program to demonstrate Node.js Functions

Line Of Code:

```
function multiply(x,y){
  return x*y;}
  let result=multiply(25,78);
  console.log(result);
```

Output Screen:

```
PS C:\Program Files\Sanyukta> Node Practical2.js
1950
PS C:\Program Files\Sanyukta>
```

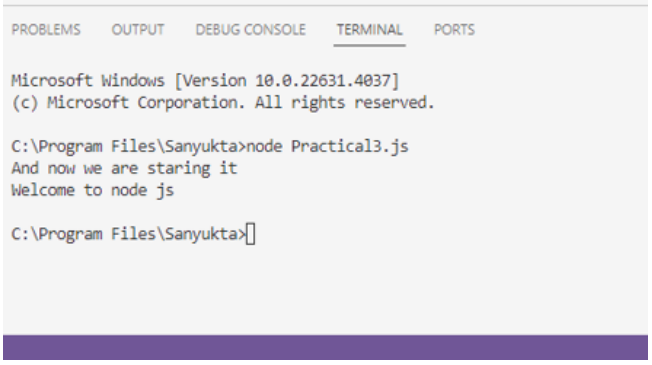
Practical 3

AIM: Write a program to demonstrate Call-back function -Anonymous function using Node JS.

Line Of Code:

```
const message=function()
 console..log("Welcome to node js");}
                setTimeout(message,10000);
                setTimeout(()=>{
                    console.log("And now we are staring it");
                },3000);
```

Output Screen:



Practical 4

AIM: Write a program to demonstrate Node.js Modules.

Line Of Code:

```
exports.myDateFun = function() {
 return new Date();
};
const dt = require('./practical4');
console.log(dt.myDateFun());
```

Output Screen:

```
Microsoft Windows [Version 10.0.22631.4169]
(c) Microsoft Corporation. All rights reserved.

C:\Program Files\Sanyukta>node pract004
2024-10-07T14:41:33.888Z
```

Practical 5

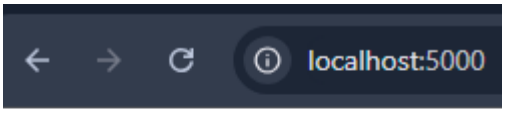
AIM: Write a program to demonstrate routing through hrrp server.

Line Of Code:

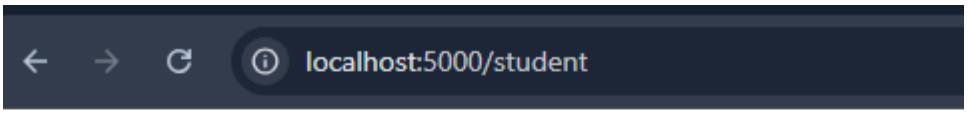
```
var http = require('http');
var server = http.createServer(function (req, res) {
    if (req.url == '/') {
        res.writeHead(200, { 'Content-Type': 'text/html' });
        res.write("<h1>Home Page</h1>");
        res.end();
    } else if (req.url == '/student') {
        res.writeHead(200, { 'Content-Type': 'text/html' });
        res.write("<h1>Master Of Computer Applications</h1>");
        res.end();
    } else if (req.url == '/admin') {
        res.writeHead(200, { 'Content-Type': 'text/html' });
        res.write("<h1>Your fee structure will be displayed on the Notice Board.</h1>");
        res.end();
    } else {
        res.writeHead(404, { 'Content-Type': 'text/html' });
        res.write("<h1>Invalid page</h1>");
        res.end();
    }});
```

Output Screen:

```
TERMINAL    PORTS

 Files\Sanyukta>node Practical5
Server is running
```



Home Page



Master Of Computer Applications

Your fee structure will be displayed on the Notice Board.

Practical 6

AIM: Write a program to demonstrate various Nodel.js Events

Line Of Code:

```
const events= require("events");
const eventEmitter= new events.EventEmitter();
function listner1(){
   console.log("Event received by Listner 1");
}
function listner2(){
   console.log("Event received by listner 2");

   eventEmitter.addListener("write",listner1);
   eventEmitter.on("write",listner2);
}
console.log(eventEmitter.listenerCount("write"));

eventEmitter.removeListener("write",listner1);
console.log("Listener 1 is removed");
eventEmitter.emit("write");

console.log(eventEmitter.listenerCount("write"));

console.log("program ended");
```

Output Screen:

```
C:\Program Files\Sanyukta>node practical6
0
Listener 1 is removed
0
program ended
```

Practical 7

AIM: Write a program to demonstrate custom event using Node JS.

Line Of Code:

```
const events=require('events');

const eventEmitter=new events.EventEmitter();

eventEmitter.on("connection",handleConnectionEvent);

function handleConnectionEvent()
{
   console.log("Connection Made !");
}
eventEmitter.emit("connection");
eventEmitter.emit("connection");
eventEmitter.emit("connection");
eventEmitter.emit("connection");

console.log("End of program");
```

Output Screen:

```
C:\Program Files\Sanyukta>node practical7
Connection Made !
Connection Made !
Connection Made !
Connection Made !
End of program
```

Practical 8

AIM: Using File Handling demonstrate all basic file operations (Create, Write, Read , Delete)

Line Of Code:
Program 1-Read

```
var fs=require('fs');

fs.readFile('write.txt',function(err,data){if(err) throw err;
console.log(data.toString());
});
```

Output Screen:

```
Microsoft Windows [Version 10.0.22631.4169]
(c) Microsoft Corporation. All rights reserved.

C:\sanyukta>node practical8
 All begin in delight and end in wisdom, as Frost taug
ht us great poems should.
```

Program 2- Write
Line Of code:

```
var fs=require('fs');
fs.writeFile('file.txt','Hi welcome to the txt file',function(err){ if(err) throw err;
else{
    console.log("Department of Master Of computer Apllication");
}
});
```

Output Screen:

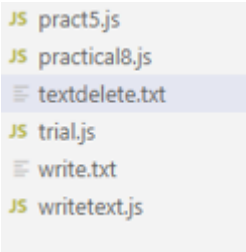
```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

PS C:\sanyukta> node writetext.js
Department of Master Of computer Apllication
PS C:\sanyukta>
```

Program 3-Delete

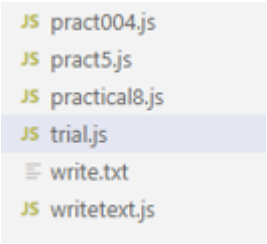
```
const fs=require('fs');
```

```
fs.unlink('textdelete.txt',function()
{
    console.log('Delete Operation Completed');
});
```

Output Screen:
Before



After Delete




```
Microsoft Windows [Version 10.0.22631.4169]
(c) Microsoft Corporation. All rights reserved.

C:\sanyukta>node Delete
Delete Operation Completed
```

Practical 9

Aim: Create an application to establish a connection with the MySQL database and perform basic database operations on it.

Line of code:

1.connection_mysql.js

```
var mysql = require('mysql');

var con = mysql.createConnection({
    host: "localhost",
    user:"root",
    password:""
});
con.connect(function(err){
    if (err) throw err;
    console.log("Connected!");
});
```

Output Screen:

```
PS C:\Users\ADMIN\Desk
Connected!
```

2.create_database

```
var mysql = require('mysql');

var con = mysql.createConnection({
    host: "localhost",
    user:"root",
    password:""
});
con.connect(function(err){
    if (err) throw err;
    console.log("Connected!");
    con.query("CREATE DATABASE Tigerdb", function (err,result){
        if (err) throw err;
        console.log("Databse created");
    });
});
```

Output Screen:

```
PS C:\Users\ADMIN\Des
Connected!
Databse created
```

3.**CREATE TABLE**

```
var mysql = require('mysql');
```

```
var con = mysql.createConnection({
    host: "localhost",
    user:"root",
    password:"",
    database: "Tigerdb"
});

con.connect(function(err){
    if (err) throw err;
    console.log("Connected!");
    var sql="CREATE TABLE customers (id INT AUTO_INCREMENT PRIMARY KEY,name
VARCHAR(255),address VARCHAR(255))";
    con.query(sql, function(err,result){
        if (err) throw err;
        console.log("Table created");
    });
});
```

Output Screen:

```
PS C:\Users\ADMIN\Des
Connected!
Table created
```

4.INSERT RECORD

```
var mysql = require('mysql');

var con = mysql.createConnection({
    host: "localhost",
    user:"root",
    password:"",
    database:"Tigerdb"
});

con.connect(function(err){
    if (err) throw err;
    console.log("Connected!");
    var sql="INSERT INTO customers(name, address) VALUES
('TOM','BOISAR'),('HARRY','PALGHAR'),('ORY','VIRAR'),('JERRY','VASAI'),('TOMMY','BOISAR')
";
    con.query(sql, function(err,result){
        if (err) throw err;
        console.log("5 RECORED INSERTED");
    });
});
```

Output Screen:

```
PS C:\Users\ADMIN\Deskto
Connected!
5 RECORED INSERTED
```

5.SELECTING RECORD

```
var mysql = require('mysql');

var con = mysql.createConnection({
    host: "localhost",
    user:"root",
    password:"",
    database: "Tigerdb"
});

con.connect(function(err){
    if (err) throw err;

con.query("SELECT * FROM customers", function(err,result, fields){
        if (err) throw err;
        console.log(result);
    });
});
```

Output Screen:

```
PS C:\Users\ADMIN\Desktop\mca\database_35> node 5selecting.js
[
  RowDataPacket { id: 2, name: 'HARRY', address: 'PALGHAR' },
  RowDataPacket { id: 3, name: 'ORY', address: 'VIRAR' },
  RowDataPacket { id: 4, name: 'JERRY', address: 'VASAI' }
]
```

6.UPDATING RECORD

```
var mysql = require('mysql');

var con = mysql.createConnection({
    host: "localhost",
    user:"root",
    password:"",
    database: "Tigerdb"
});
con.connect(function(err){
    if (err) throw err;
    var sql="UPDATE customers SET address = 'MUMBAI' WHERE address = 'BOISAR'";
    con.query(sql, function(err,result){
        if (err) throw err;
        console.log(result.affectedRows + "record(s) updated");
    });
});
```

Output Screen:

```
PS C:\Users\ADMIN\Desktop\mca\d
2record(s) updated
```

6.DELETING RECORD

```
var mysql = require('mysql');
```

```
var con = mysql.createConnection({
    host: "localhost",
    user:"root",
    password:"",
    database: "Tigerdb"
});
con.connect(function(err){
    if (err) throw err;
    var sql="DELETE FROM customers WHERE address = 'MUMBAI'";
    con.query(sql, function(err,result){
        if (err) throw err;
        console.log("Number of records deleted: "+ result.affectedRows);
    });
});
```

Output Screen:

```
PS C:\Users\ADMIN\Desktop\mca\data
Number of records deleted: 2
```

Practical 10

Aim:Created the application with react js to implement the component lifecycle

Source code:

```
import React, { Component } from 'react'; // Import React and Component
import './App.css'; // Import your CSS file
class LifeCycleDemo extends Component {
  constructor(props) {
    super(props);
    this.state = { counter: 0 };
    console.log('Constructor: Initializing state');
  }
  componentDidMount() {
    console.log('Component Did Mount: Component has been mounted in the DOM');
  }
  componentDidUpdate(prevProps, prevState) {
    if (prevState.counter !== this.state.counter) {
      console.log('Component Did Update: State has changed, re-rendered');
    }
  }
  componentWillUnmount() {
    console.log('Component Will Unmount: Component is about to be removed');
  }
  increaseCounter = () => {
    this.setState({ counter: this.state.counter + 1 });
  };
  render() {
    console.log('Render: Rendering the component');
    return (
      <div>
        <h1>React Component Life Cycle</h1>
        <p>Counter: {this.state.counter}</p>
        <button onClick={this.increaseCounter}>Increase Counter</button>
      </div>
    );}}
export default LifeCycleDemo;
```

output

Practical 11

Aim: **Create an application to implement class and functional components in ReactJS.**

Source code:

Step 1: **Setting Up the React App**

1. Create React App:

npx create-react-app react-component-demo

cd react-component-demo
npm start

Step 2: Creating Class and Functional Components

We'll create two separate components, one using the class syntax and one using functional syntax.

2.1 Class Component

In src/ClassComponent.js, create a class component:

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

class ClassComponent extends Component {
 constructor(props) {
 super(props);
 this.state = {
 message: 'Hello from Class Component!',
 };
 }

 render() {
 return (
 <div style={{ border: '2px solid blue', padding: '20px', margin: '10px'
}}>  <h2>{this.state.message}</h2>
 <p>This is rendered using a class component.</p>
 </div>
 );
 }
}
export default ClassComponent;
```

2.2 Functional Component

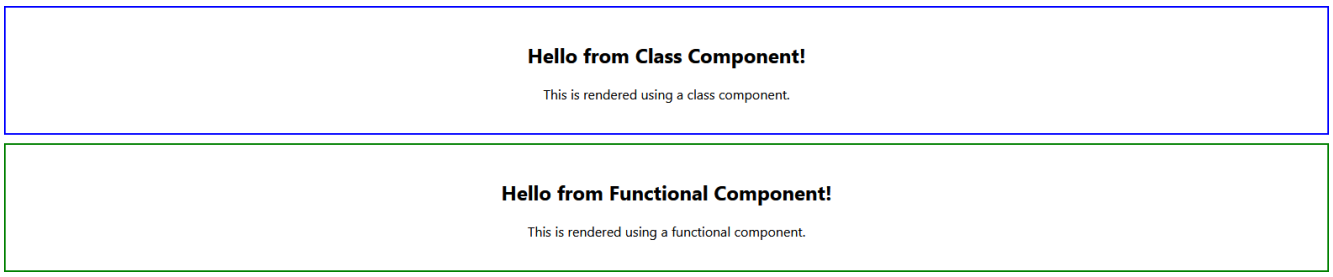
In src/FunctionalComponent.js, create a functional component:

```
import React, { useState } from 'react';
function FunctionalComponent() {
 const [message] = useState('Hello from Functional Component!');
 return (
 <div style={{ border: '2px solid green', padding: '20px', margin: '10px'
}}>  <h2>{message}</h2>
 <p>This is rendered using a functional component.</p>  </div>
 );
}

export default FunctionalComponent;
```

2.3: Do this changes in App,js

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

        <ClassComponent/>
        <FunctionalComponent/>

    </div>
  );
}
```

Output:



Practical 12

Aim: **Create an application in ReactJS to import and export components.Source code:**

Step 1: Setting Up the React App

```
npx create-react-app react-import-export-demo
cd react-import-export-demo
npm start
```

Step 2: Create Multiple Components

We'll create three components, each in separate files, and import them into the main App.js file.

2.1 Header Component
Header.js

```
import React from 'react';
function Header() {
return (
<header style={{ backgroundColor: '#4CAF50', color: 'white', padding: '10px' }}>
<h1>Welcome to React Import and Export Demo</h1>
</header>
);
}
export default Header;
```

2.2 Footer Component
Footer.js

```
import React from 'react';
function Footer()  {
return (
<footer style={{ backgroundColor: '#333', color: 'white', padding: '10px',
position: 'fixed', bottom: 0, width: '100%' }}>
<p>React Import and Export Demo © 2024</p>
</footer>
);
}
export default Footer;
```

2.3 MainContent Component
MainComponent.js

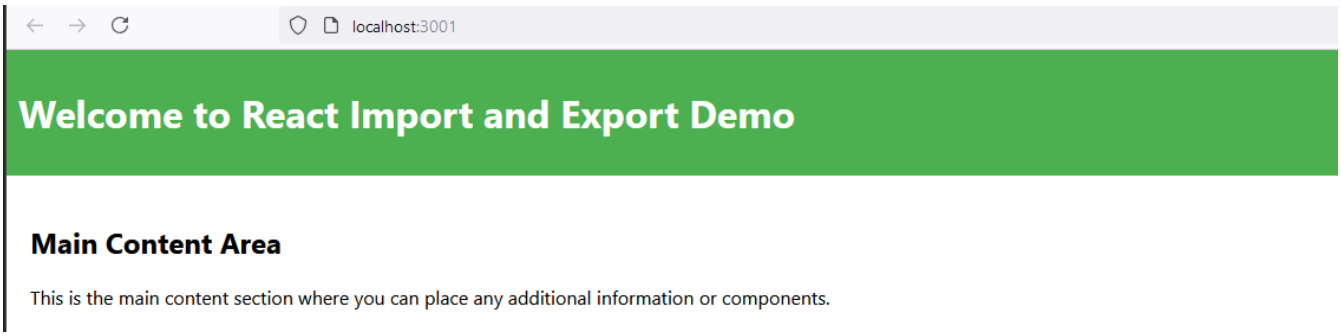
```
import React from 'react';
import Header from './Header'; // Importing Header component
import MainContent from './MainContent'; // Importing MainContent component
import Footer from './Footer'; // Importing Footer component
```

```
function App() {
return (
<div>
<Header />
<MainContent />
<Footer />
</div>
);
}
export default App;
```

Step 3: Import Components into App

```
import React from 'react';
import Header from './Header'; // Importing Header component
import MainContent from './MainContent'; // Importing MainContent component
import Footer from './Footer'; // Importing Footer component

function App() {
return (
<div>
<Header />
<MainContent />
<Footer />
</div>
);
}
export default App;
```

Output:



Practical 13

Aim: Create an application to implement state and props in ReactJS.

Source code:
Step 1: Setting Up the React App
npx create-react-app react-state-props-demo
cd react-state-props-demo
npm start

2.1 Parent Component

```
; import React, { useState } from 'react';
import ChildComponent from './ChildComponent';
function ParentComponent() {
 // Define state
 const [message, setMessage] = useState('Hello from Parent!');
 const updateMessage = () => {
 setMessage('I LOVE PANIPURI!!');
 };
 return (
 <div style={{ border: '2px solid blue', padding: '20px', margin: '10px' }}>  <h2>Parent
Component</h2>
 <p>Message in Parent: {message}</p>
 {/* Passing message and function as props to ChildComponent */}  <ChildComponent
message={message} updateMessage={updateMessage} />  </div>
 );
}
export default ParentComponent;
```

Step 3: Create a Child Component

```
import React from 'react';
function ChildComponent({ message, updateMessage }) {
return (
<div style={{ border: '2px solid green', padding: '20px', margin: '10px' }}>
<h2>Child Component</h2>
<p>Message from Parent: {message}</p>
<button onClick={updateMessage}>CLICK HERE FOR MORE INFO</button>
</div>
);
}
export default ChildComponent;
```

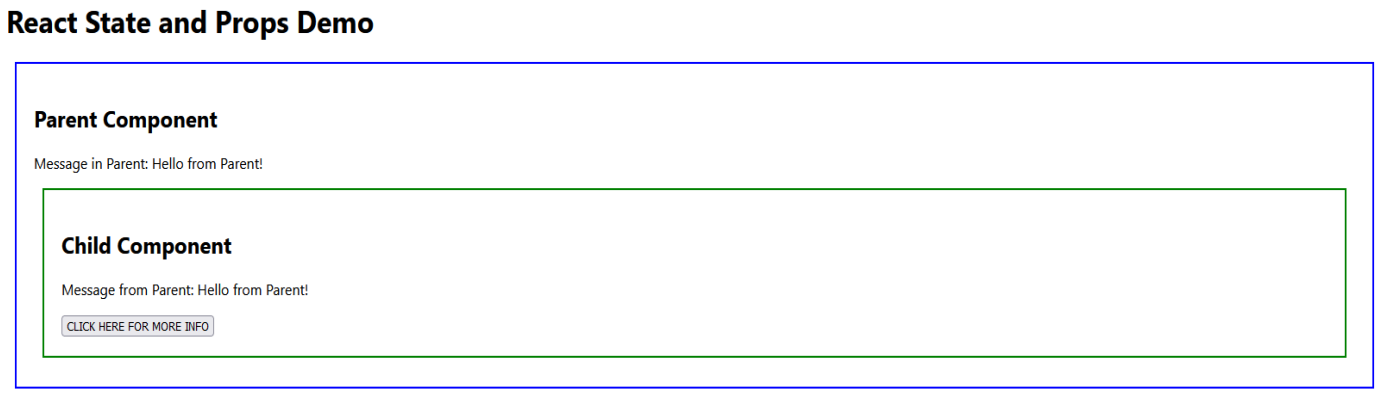
4.1 Update App.js

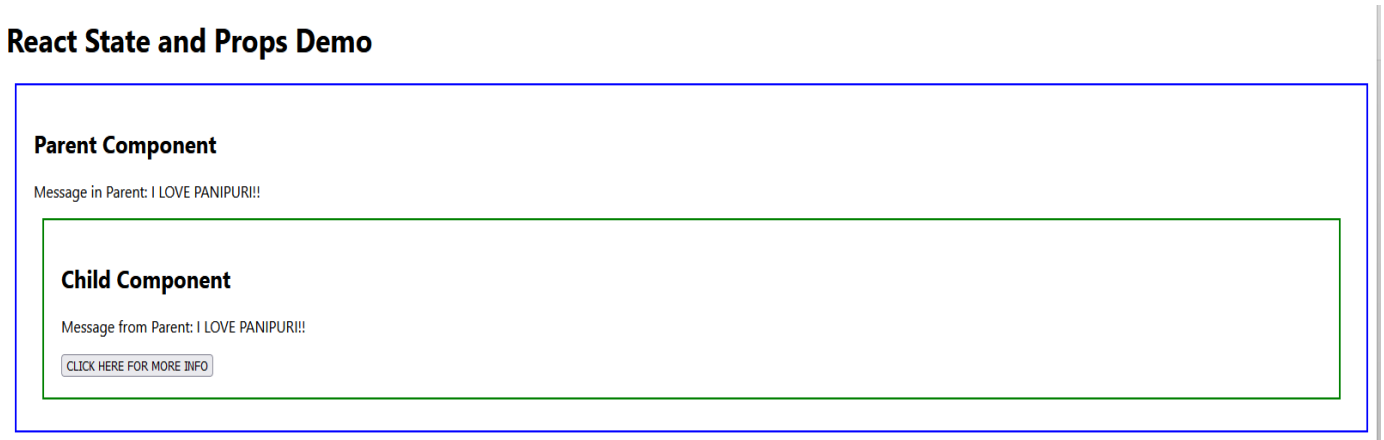
Open src/App.js and modify it as follows:

```
import React from 'react';
import ParentComponent from './ParentComponent';
function App() {
```

```
return (
<div className="App">
<h1>React State and Props Demo</h1>
<ParentComponent />
</div>
);
}
export default App;
```

Output:





Practical 14

Aim: Create an application in ReactJS to use DOM events.
Source code:

Step 1: Setting Up the React App
npx create-react-app react-dom-events-demo
cd react-dom-events-demo
npm start

Step 2: Handling Different DOM Events
2.1 Create a DOMEventsComponent

DOMEventsComponent.js

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

function DOMEventsComponent() {
 const [inputValue, setInputValue] = useState('');
 const [hovered, setHovered] = useState(false);

 // Handle button click
 const handleClick = () => {
 alert('Button clicked!');
 };

 // Handle input change
 const handleChange = (event) => {
 setInputValue(event.target.value);
 };

 // Handle mouse hover
 const handleMouseOver = () => {
 setHovered(true);
 };

 const handleMouseOut = () => {
 setHovered(false);
 };

 return (
 <div style={{ padding: '20px', textAlign: 'center' }}>
 <h2>React DOM Events Demo</h2>

 {/* onClick Event */}
 <button onClick={handleClick} style={{ padding: '10px', fontSize: '16px'
}}>  Click Me
 </button>
```

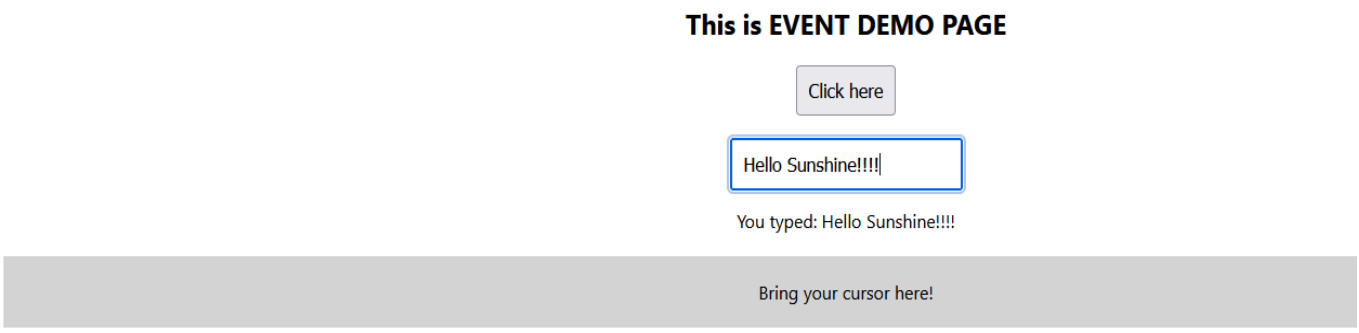
```
 {/* onChange Event */}
 <div style={{ margin: '20px 0' }}>
 <input
 type="text"
 placeholder="Type something..."
 value={inputValue}
 onChange={handleChange}
 style={{ padding: '10px', fontSize: '16px' }}
 />
 <p>You typed: {inputValue}</p>
 </div>

 {/* onMouseOver Event */}
 <div
 onMouseOver={handleMouseOver}
 onMouseOut={handleMouseOut}
 style={{
 backgroundColor: hovered ? 'lightblue' : 'lightgray',
 padding: '20px',
 cursor: 'pointer',
 }}
 >
 {hovered ? 'Mouse is Over' : 'Hover over this box'}
 </div>
 </div>
 );
}

export default DOMEventsComponent;
```

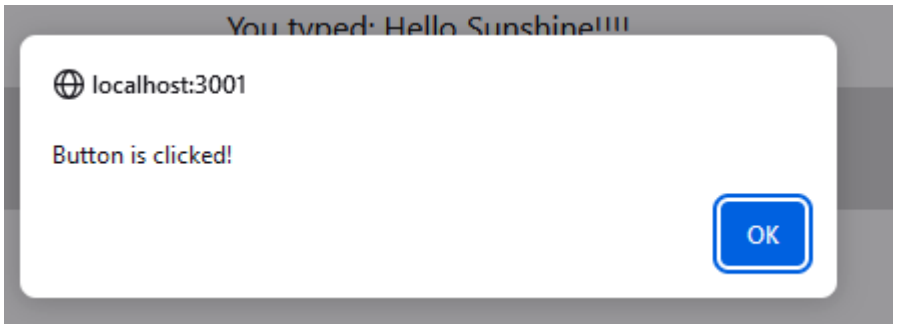
Step 3: Modify App Component
3.1 Update App.js
Open src/App.js and modify it as follows:

```
import React from 'react';
import DOMEventsComponent from './DOMEventsComponent';

function App() {
 return (
 <div className="App">
 <h1>React DOM Events Example</h1>
 <DOMEventsComponent />
 </div>
 );
}

export default App;
```

Ouput:

DOM Events Example





Practical 15

Aim: Create an application in ReactJS form and add client and server-side validation.

Source code:

Step 1: Setting Up the React App

npx create-react-app react-form-validation-demo
cd react-form-validation-demo
npm start

Step 2: Create the Form Component

2.1 Create a RegistrationForm.js

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

function RegistrationForm() {
 const [formData, setFormData] = useState({
 name: '',
 email: '',
 password: ''
 });

 const [formErrors, setFormErrors] = useState({});
 const [isSubmitted, setIsSubmitted] = useState(false);

 // Handle form input changes
 const handleChange = (e) => {
 const { name, value } = e.target;
 setFormData({
 ...formData,
 [name]: value
 });
 };

 // Validate form data
 const validate = () => {
 let errors = {};
 if (!formData.name) errors.name = 'Name is required';
 if (!formData.email) {
 errors.email = 'Email is required';
 } else if (!/\S+@\S+\.\S+/.test(formData.email)) {
 errors.email = 'Email address is invalid';
 }
 if (!formData.password) {
 errors.password = 'Password is required';
```

```
} else if (formData.password.length < 6) {
errors.password = 'Password must be at least 6 characters';  }
return errors;
};

// Handle form submission
const handleSubmit = (e) => {
e.preventDefault();
const errors = validate();
setFormErrors(errors);

if (Object.keys(errors).length === 0) {
// Mock server request
setIsSubmitted(true);
console.log('Form data submitted:', formData);
// In real implementation, send data to the server here
} else {
setIsSubmitted(false);
}
};

return (
<div className="form-container" style={{ textAlign: 'left', padding: '20px'
}}>  <h2>Registration Form</h2>
{isSubmitted && <p style={{ color: 'green' }}>Form
submitted  successfully!</p>}
<form onSubmit={handleSubmit}>
<div>
<label>Name:</label>
<input
type="text"
name="name"
value={formData.name}
onChange={handleChange}
style={{ display: 'block', marginBottom: '10px', padding: '5px' }}  />
{formErrors.name && <p style={{ color: 'red' }}>{formErrors.name}</p>}  </div>

<div>
<label>Email:</label>
<input
type="email"
name="email"
value={formData.email}
onChange={handleChange}
style={{ display: 'block', marginBottom: '10px', padding: '5px' }}  />
{formErrors.email && <p style={{ color: 'red' }}>{formErrors.email}</p>}  </div>

<div>
<label>Password:</label>
```

```
 <input
 type="password"
 name="password"
 value={formData.password}
 onChange={handleChange}
 style={{ display: 'block', marginBottom: '10px', padding: '5px' }}  />
 {formErrors.password && <p style={{ color: 'red'
}}>{formErrors.password}</p>}
 </div>
 <button type="submit" style={{ padding: '10px 20px', marginTop: '10px' }}>  Submit
 </button>
 </form>
 </div>
 );
}
export default RegistrationForm;
```

Step 3: Modify App Component
3.1 Update App.js

```
import React from 'react';
import RegistrationForm from './RegistrationForm';
function App() {
 return (
 <div className="App">
 <h1>React Form with Validation</h1>
 <RegistrationForm />
 </div>
 );
}
export default App;
```

Step 4: Running the App

npm start

Step 5: Adding Server-Side Validation (Optional)

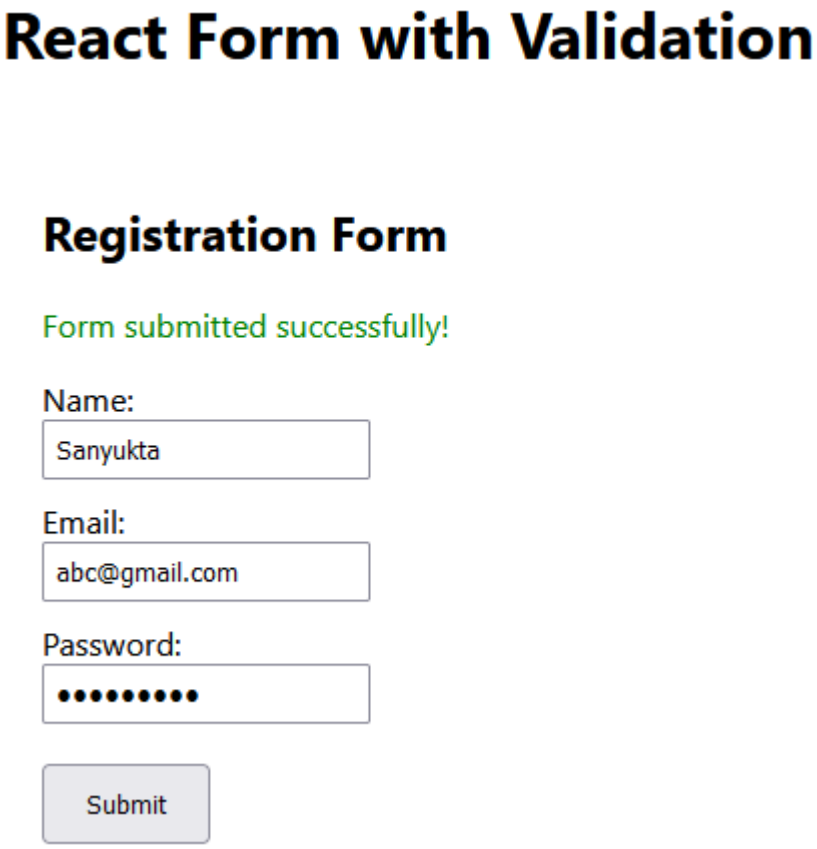
For server-side validation, we will mock a simple server using an API call. Normally, you would send form data to an actual server to check its validity. You can use a backend framework like Node.js or Django for that. To simulate a server response, we can create a mock API call using fetch or any other HTTP client.

5.1 Mock Server Validation
Modify the handleSubmit function to mock a server request:

```
// Handle form submission with mock server request
const handleSubmit = async (e) => {
 e.preventDefault();
 const errors = validate();
```

```
setFormErrors(errors);

if (Object.keys(errors).length === 0) {
try {
const response = await fetch('https://jsonplaceholder.typicode.com/posts', {  method:
'POST',
body: JSON.stringify(formData),
headers: {
'Content-type': 'application/json; charset=UTF-8',
},
});
if (response.ok) {
setIsSubmitted(true);
console.log('Form data sent to the server:', formData);
}
} catch (error) {
console.error('Server validation failed', error);
}
} else {
setIsSubmitted(false);
}
};
```

Output:



Practical 16

Aim: Create an application in ReactJS that uses routing for navigation..
Source code:

Step 1: Setting Up the React App

npx create-react-app react-routing-demo
cd react-routing-demo
npm install react-router-dom
npm start

Step 2: Setting Up React Router

2.1 Create the Components

Home.js

```
import React from 'react';

function Home() {
 return (
 <div style={{ textAlign: 'center', padding: '20px' }}>
 <h2>Welcome to the Home Page</h2>
 <p>This is the main page of our application.</p>
 </div>
 );
}
export default Home;
```

About.js

```
import React from 'react';

function About() {
 return (
 <div style={{ textAlign: 'center', padding: '20px' }}>
 <h2>About Us</h2>
 <p>This is the about page where we describe our app.</p>  </div>
 );
}
export default About;
```

Contact.js

```
import React from 'react';

function Contact() {
 return (
 <div style={{ textAlign: 'center', padding: '20px' }}>
 <h2>Contact Us</h2>
 <p>This is the contact page for inquiries.</p>
 </div>
 );
}
export default Contact;
```

Step 3: Setting Up the Router

3.1 Update App.js

```
import React from 'react';
import { BrowserRouter as Router, Route, Routes, Link } from 'react-router-dom'; import
Home from './Home';
import About from './About';
import Contact from './Contact';
function App() {
 return (
 <Router>
 <div style={{ textAlign: 'center', padding: '20px' }}>
 <h1>React Routing Demo</h1>

 {/* Navigation Links */}
 <nav style={{ marginBottom: '20px' }}>
 <Link to="/" style={{ margin: '0 15px' }}>Home</Link>  <Link to="/about"
style={{ margin: '0 15px' }}>About</Link>  <Link to="/contact" style={{
margin: '0 15px' }}>Contact</Link>  </nav>

 {/* Route Definitions */}
 <Routes>
 <Route path="/" element={<Home />} />
 <Route path="/about" element={<About />} />
 <Route path="/contact" element={<Contact />} />
 </Routes>
 </div>
 </Router>
 );
}
export default App;
```

Output:

React Routing Demo

Home About Contact

Welcome to the Home Page

This is the main page of our application.

React Routing Demo

Home About Contact

About Us

This is the about page where we describe our app.

React Routing Demo

Home About Contact

Contact Us

This is the contact page for inquiries.

Practical 17

Aim: Create a simple ReactJS application with Hooks (useState, useEffect, useContext).

Source Code:

Counter.js

```
import React, { useState } from 'react';
function Counter() {
const [count, setCount] = useState(0);
return (
<div style={{ textAlign: 'center', padding: '20px' }}>
<h2>Counter</h2>
<p>Current Count: {count}</p>
<button onClick={() => setCount(count + 1)}>Increase</button>
<button onClick={() => setCount(count - 1)} style={{ marginLeft: '10px'
}}>Decrease</button>
</div>
);
}
export default Counter;
```

DataFetching.js

```
import React, { useState, useEffect } from 'react';
function DataFetching() {
const [data, setData] = useState([]);
const [loading, setLoading] = useState(true);
useEffect(() => {
fetch('https://jsonplaceholder.typicode.com/posts')
.then((response) => response.json())
.then((data) => {
setData(data.slice(0, 5)); // Displaying only the first 5 items
setLoading(false);
})
.catch((error) => console.error(error));
}, []); // Empty array ensures this runs once on mount
return (
<div style={{ textAlign: 'center', padding: '20px' }}>
<h2>Data Fetching</h2>
{loading ? (
<p>Loading...</p>
) : (
<ul>
{data.map((item) => (
<li key={item.id}>{item.title}</li>
))}
</ul>
)}
```

```
</div>
);
}
export default DataFetching;
```

ThemeContext.js

```
import React, { createContext, useState } from 'react';
export const ThemeContext = createContext();
export const ThemeProvider = ({ children }) => {
const [isDarkTheme, setIsDarkTheme] = useState(false);
const toggleTheme = () => {
setIsDarkTheme((prevTheme) => !prevTheme);
};return (
<ThemeContext.Provider value={{ isDarkTheme, toggleTheme }}>
{children}
</ThemeContext.Provider>
);
};
```

ThemedComponent.js

```
import React, { useContext } from 'react';
import { ThemeContext } from './ThemeContext';
function ThemedComponent() {
const { isDarkTheme, toggleTheme } = useContext(ThemeContext);
return (
<div>style={{
textAlign: 'center',
padding: '20px',
backgroundColor: isDarkTheme ? '#333' : '#fff',
color: isDarkTheme ? '#fff' : '#000',
}}
>
<h2>Theme Toggle</h2>
<p>Current Theme: {isDarkTheme ? 'Dark' : 'Light'}</p>
<button onClick={toggleTheme}>Toggle Theme</button>
</div>);
}
export default ThemedComponent;
```

App.js

```
import React from 'react';
import Counter from './Counter';
import DataFetching from './DataFetching';
import ThemedComponent from './ThemedComponent';
import { ThemeProvider } from './ThemeContext';
function App() {
return (
<ThemeProvider>
<div className="App" style={{ textAlign: 'center', padding: '20px' }}>
```

```
<h1>React Hooks Demo</h1>
<Counter />
<DataFetching />
<ThemedComponent />
</div>
</ThemeProvider>
);}
export default App;
```

Output: