Roll No :- 48
Name:- Manav Jain
Date:-25/08/2023

ASSIGNMENT – 6(b)

AIM:-WAP to implement the concept of forms and events. Create a React JS registration Form consisting of textbox, textarea, selection input, check box, radio button, submit button and reset button handling onSubmit, onClick and keyDown events.

LO MAPPED:- LO5

THEORY:-

A React component is a building block of a React application. It encapsulates a part of the user interface and its logic. In the case of a registration form, you would typically create a custom component called RegistrationForm to encapsulate the form's structure and behavior.

A registration form consists of various HTML form elements like text inputs, textareas, select dropdowns, checkboxes, and radio buttons. In React, you can use JSX to define these form elements within the render method of your component. Each form element should be associated with a piece of state.

The components are:

1) **Textbox (Input):** The text input field where users can enter text. The value attribute is controlled by the component's state, and the onChange event handler (handleInputChange) is triggered whenever the user types something in the input field. It updates the name field in the component's state.

e.g: <input
type="text"
name="name"
value={this.state.name}
onChange={this.handleInputChange} />

2) **TextArea:** The textarea element allows users to input multiple lines of text. Similar to the textbox, the value attribute is controlled by the component's state, and the onChange event handler updates the bio field in the state as the user types.

e.g: <textarea
name="bio"
value={this.state.bio}
onChange={this.handleInputChange}
/>

3) **Selection Input:** The select element creates a dropdown menu where users can select an option. The value attribute is controlled by the component's state, and the onChange event handler (handleInputChange) updates the gender field in the state when the user selects a different option.

e.g: <select
name="gender"
value={this.state.gender}
onChange={this.handleInputChange}
>
<option value="male">Male</option>
<option value="female">Female</option>
<option value="other">Other</option>
</select>

4) **Checkbox:** It creates a checkbox input that allows users to toggle a boolean value (agree) in the component's state. The checked attribute is controlled by the state, and the onChange event handler updates the agree field in the state when the user checks or unchecks the checkbox.

e.g: <label>
<input
type="checkbox"
name="agree"
checked={this.state.agree}
onChange={this.handleInputChange}
/>

```
 I agree to the terms and conditions
</label>
```

5) **Radio Buttons:** Radio buttons allow users to choose one option from a set of options. They share the same name attribute to create a group. The checked attribute is controlled by the state, and the onChange event handler updates the selectedOption field in the state when the user selects one of the radio buttons.

e.g:

```
<label>
 <input
  type="radio"
  name="selectedOption"
  value="option1"
  checked={this.state.selectedOption === 'option1'}
  onChange={this.handleInputChange}
 />
 Option 1
</label>
<label>
 <input
  type="radio"
  name="selectedOption"
  value="option2"
  checked={this.state.selectedOption === 'option2'}
  onChange={this.handleInputChange}
 />
 Option 2
</label>
```

6) **Submit Button:** This button triggers the form submission when clicked. It doesn't need an onClick handler because the form's onSubmit event is used to handle the form submission.

e.g:

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

7) **Reset Button**: The reset button is used to clear the form fields. It has an onClick event handler (handleReset) that resets the component's state when clicked.

e.g: <button type="button" onClick={this.handleReset}>Reset</button>

8) **KeyDown Event (Example with an input field):** The onKeyDown event allows you to capture keyboard events. In this example, it's used with a text input field. When the user presses a key, the handleKeyDown event handler is called. In this case, it checks if the Enter key was pressed and logs a message.These elements and event handlers work together to create a functional registration form in ReactJS. They demonstrate how to handle user input, form submission, and keyboard events in a React application

e.g: **<input**
type="text"
placeholder="Press Enter Key"
onKeyDown={this.handleKeyDown}
/>

OUTPUT WITH CODE:

1)Index.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>Registration Form</title>
</head>
<body>
  <div id="root"></div>
</body>
</html>
```

2)Registration_Form.jsx:

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

```
import './regformst.css';
class RegistrationForm extends Component {
  constructor() {
    super();
    this.state = {
      name: '',
      email: '',
      gender: 'male',
      country: 'india',
      bio: '',
      agreement: false,
    };
  }

  handleInputChange = (event) => {
    const { name, value, type, checked } = event.target;

    // Handle checkbox input differently
    if (type === 'checkbox') {
      this.setState({ [name]: checked });
    } else {
      this.setState({ [name]: value });
    }
  };

  handleSubmit = (event) => {
    event.preventDefault();
    // You can perform form submission logic here
    if(this.state.name=== "" || this.state.email=== "" || this.state.bio=== "")
      {
        alert('please enter the data first');
      }
      else
```

```
    {
      alert('Form submitted with data:\nName: ' + this.state.name + '\nEmail id: ' +
this.state.email + '\nGender: ' + this.state.gender + '\nCountry: ' +
this.state.country + '\nBio: ' + this.state.bio + '\n');



    }
  };

  handleReset = () => {
    this.setState({
      name: '',
      email: '',
      gender: 'male',
      country: 'india',
      bio: '',
      agreement: false,
    });
  };

  handleKeyDown = (event) => {
    if (event.key === 'Enter') {
      this.handleSubmit(event);
    }
  };

  render() {
    return (
      <form onSubmit={this.handleSubmit} className="registration-form">
        <div className="form-group">
          <label>Name:</label>
          <input
            type="text"
```

```
    name="name"
    value={this.state.name}
    onChange={this.handleInputChange}
    onKeyDown={this.handleKeyDown}
    required
   />
  </div>
  <div className="form-group">
   <label>Email:</label>
   <input
    type="email"
    name="email"
    value={this.state.email}
    onChange={this.handleInputChange}
    onKeyDown={this.handleKeyDown}
    required
   />
  </div>
  <div className="form-group">
   <label>Gender:</label>
   <div>
    <label>
     <input
      type="radio"
      name="gender"
      value="male"
      checked={this.state.gender === 'male'}
      onChange={this.handleInputChange}
     />{' '}
     Male
    </label>
    <label>
     <input
```

```
        type="radio"
        name="gender"
        value="female"
        checked={this.state.gender === 'female'}
        onChange={this.handleInputChange}
      />{' '}
      Female
    </label>
    <label>
      <input
        type="radio"
        name="gender"
        value="other"
        checked={this.state.gender === 'other'}
        onChange={this.handleInputChange}
      />{' '}
      Other
    </label>
  </div>
</div>
<div className="form-group">
  <label>Country:</label>
  <select
    name="country"
    value={this.state.country}
    onChange={this.handleInputChange}
  >
    <option value="india">India</option>
    <option value="usa">United States</option>
    <option value="canada">Canada</option>
    <option value="uk">United Kingdom</option>
    <option value="australia">Australia</option>
  </select>
```

```
        </div>
        <div className="form-group">
          <label>Bio:</label>
          <textarea
            name="bio"
            value={this.state.bio}
            onChange={this.handleInputChange}
            onKeyDown={this.handleKeyDown}
          />
        </div>
        <div className="form-group">
          <label>
            <input
              type="checkbox"
              name="agreement"
              checked={this.state.agreement}
              onChange={this.handleInputChange}
            />{' '}
            I agree to the terms and conditions
          </label>
        </div>
        <div className="form-group">
          <button type="submit" onClick={this.handleSubmit}>
            Submit
          </button>
          <button type="button" onClick={this.handleReset}>
            Reset
          </button>
        </div>
      </form>
    );
  }
}
```

```
export default RegistrationForm;
```

3)regformst.css:

```
form {
  max-width: 400px;
  margin: 40px auto;
  padding: 20px;
  border: 1px solid #ccc;
  border-radius: 10px;
  box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
  background-color: #f0f0f0; /* Change the background color */
  color: #333; /* Change the text color */
  font-family: Arial, sans-serif; /* Change the font family */
  font-size: 18px; /* Change the font size */
}

h1 {
  text-align: center;
  color: #555; /* Change the heading text color */
}

div {
  margin: 15px;
}

label {
  display: block;
  font-weight: bold;
  margin-bottom: 5px;
  color: #666; /* Change label text color */
}
```

```
input[type="text"],
input[type="email"],
select,
textarea {
  width: 100%;
  padding: 10px;
  border: 1px solid #ccc;
  border-radius: 4px;
  font-size: 16px; /* Change the input font size */
}

textarea {
  resize: vertical;
}

input[type="checkbox"] {
  margin-right: 5px;
}

button {
  padding: 10px 20px;
  background-color: #5a027d; /* Change the button background color */
  color: #fff;
  border: none;
  border-radius: 4px;
  cursor: pointer;
  font-size: 18px; /* Change the button font size */
  font-weight: bold; /* Add font weight */
}

button[type="button"] {
  background-color: #ff00ae;
  margin-left: 10px;
```

```
}

button:hover {
  background-color: #0056b3;
}

h2 {
  text-align: center;
}
```

4)App.js:

```
import React from 'react';
import './App.css';
import RegistrationForm from './components/Registration_Form';

function App() {
  return (
    <div className="App">
      <h1>Registration Form</h1>
      <RegistrationForm />
    </div>
  );
}

export default App;
```

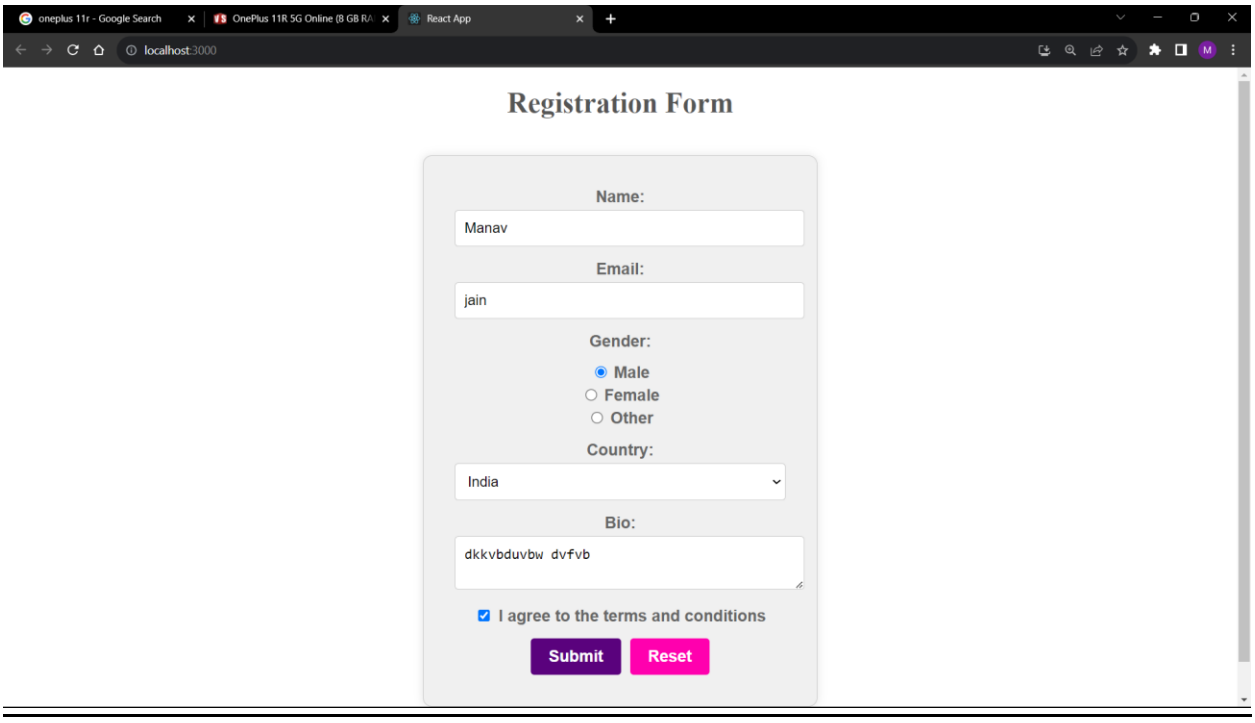
5)Index.js:

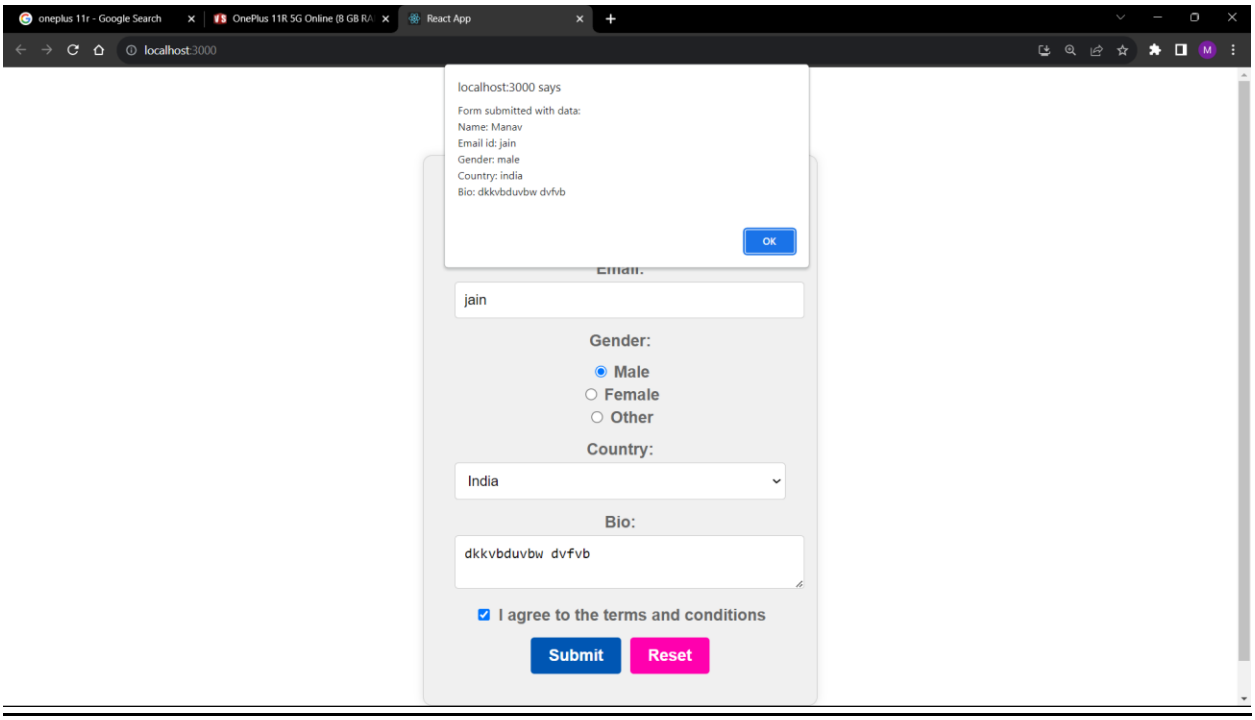
```
import React from 'react';
import ReactDOM from 'react-dom';
import App from './App';

ReactDOM.render(
```

```
  <React.StrictMode>
    <App />
  </React.StrictMode>,
  document.getElementById('root')
);
```

Output:





CONCLUSION:- In ReactJS, form elements function uniquely compared to other DOM elements because they inherently manage their own state. React's event system, called Synthetic Events, provides a consistent way to handle events across browsers. I've successfully built a registration form in React using form elements and effectively managed events like KeyDown for keyboard input, onSubmit for form submission, and onClick for button clicks.