Assignment 6b

Roll No: 71 Batch T14

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.

Theory:

Forms in ReactJS:

1. Controlled Components:
   In React, form elements like input fields, textareas, and select boxes are turned into "controlled components." This means that their values are controlled by the component's state. You store the input value in the component's state and update it as the user interacts with the form.
2. State Management:
   React components maintain their state, which includes form data. When the user interacts with the form, React updates the component state, and this, in turn, updates the form elements, ensuring that they always display the correct values.
3. Event Handling:
   React provides a consistent way to handle events, such as clicks, form submissions, and keypresses. You can attach event handlers to specific elements or components to respond to user interactions.

Event Handling in ReactJS:

1. Event Binding:
   You can bind event handlers to React components using the onClick, onChange, onSubmit, and similar event attributes. These event handlers are functions that get executed when a specific event occurs.
2. Event Object:
   When an event occurs, React passes an event object to the corresponding event handler. This object contains information about the event, such as the target element and event type. You can access this object to gather more details about the event.
3. Preventing Default Behavior:
   You can use the preventDefault method of the event object to prevent the default behavior of certain events. For example, you can prevent a form from submitting when handling the onSubmit event, allowing you to perform custom validation or data processing.
4. Updating State:
   Event handlers often involve updating the component's state. By calling setState(), you can modify the state data, which triggers a re-render of the component and updates the user interface to reflect the new state.
5. Passing Data:
   You can pass data from the event handler to other parts of your application or to child components using React's props system. This enables you to share information between components efficiently.

Form components:

1. Textbox (<input type="text">):
- A textbox is a standard input field where users can enter text information, such as their name.
- Event Handling: The onChange event is commonly used with textboxes to capture changes in the input value as the user types.

2. Textarea (<textarea>):
- A textarea is a multi-line input field used for longer text entries, like an address or comments.
- Event Handling: Similar to textboxes, you can use the onChange event to track changes in the textarea's content.

3. Selection Input (<select>) with Options:
- A selection input, often called a dropdown or select box, allows users to choose from a list of predefined options.
- Event Handling: The onChange event is used here as well to detect when the user selects an option.

4. Checkbox (<input type="checkbox">):
- A checkbox is used for binary choices, such as subscribing to a newsletter.
- Event Handling: The onChange event can be used to monitor whether the checkbox is checked or unchecked.

5. Radio Buttons (<input type="radio">):
- Radio buttons are used when there are multiple mutually exclusive choices (e.g., gender).
- Event Handling: Like checkboxes, radio buttons also use the onChange event to determine which option the user has selected.

6. Submit Button (<button type="submit">):

- The submit button is used to send the form data to the server for processing.

- Event Handling: The onSubmit event is associated with the form element and is triggered when the user clicks the submit button. It's used to handle form submission logic, such as data validation and sending data to the server.

7. Reset Button (<button type="reset">):

- The reset button is used to clear or reset the form's input fields to their initial values.

- Event Handling: The onClick event is used with the reset button to trigger a function that clears the form fields, returning them to their default or empty state.

8. Keydown Event:

- The keydown event occurs when a key on the keyboard is pressed down.

- While not typically associated with form submission, you can use the keydown event to capture specific keyboard inputs (e.g., Enter key) to trigger custom actions, such as submitting the form when the Enter key is pressed.

Conclusion:

Understood the concept of event and forms in React.js and implemented in a program which create a form.

LO5 was achieved.

Created a form consisting of textbox, textarea, selection input, check box, radio button, submit button and reset button handling onSubmit, onClick and keyDown events