React Forms and Controlled Components

# 1. Explain about React Forms

React forms are used to capture user input from the interface. They work similarly to HTML forms but with a key difference—form elements in React can be controlled by the component's state.  
Instead of the browser handling the form data, React components handle the input data via the useState hook and event handlers. This enables validation, dynamic form behavior, and better data management.

# 2. Define Controlled Components

A controlled component is an input form element where the value is controlled by React using state. In a controlled component, form data is handled by the state of a component.  
This means every time the user types in an input field, an event handler updates the state, and the component re-renders with the new value.  
  
Example:  
const [name, setName] = useState("");  
<input type="text" value={name} onChange={(e) => setName(e.target.value)} />

# 3. Explain about Various Input Controls

In React, various input controls are available and used similarly to traditional HTML inputs, but they are often controlled via state.  
  
- Textbox (`<input type="text">`) – Used to capture text data.  
- Textarea (`<textarea>`) – Used for longer, multiline input.  
- Button (`<button>`) – Used to trigger form submission or actions.  
- Radio buttons (`<input type="radio">`) – Used to select one option from a group.  
- Checkboxes (`<input type="checkbox">`) – Used for multiple selections.  
- Dropdowns (`<select>`) – Used to select from a list of items.  
All these controls can be managed using React state and the onChange event.

# 4. Explain about Handling Forms

Handling forms in React involves managing the form data using state and handling events such as onChange and onSubmit.  
  
Steps to handle forms:  
1. Use the useState hook to create state variables for each form input.  
2. Add onChange handlers to each input field to update the corresponding state.  
3. Add an onSubmit handler on the form to capture submission events.  
  
Example:  
const [email, setEmail] = useState("");  
<input type="email" value={email} onChange={(e) => setEmail(e.target.value)} />

# 5. Explain about Submitting Forms

Form submission in React is handled using the onSubmit event on the form element. To prevent the default behavior of refreshing the page, you use event.preventDefault().  
  
Typical steps:  
1. Attach an onSubmit handler to the form.  
2. Inside the handler, prevent default behavior.  
3. Process the state variables as needed (e.g., send them to a server or display them).  
  
Example:  
const handleSubmit = (e) => {  
 e.preventDefault();  
 alert("Form submitted!");  
};  
  
<form onSubmit={handleSubmit}>  
 <button type="submit">Submit</button>  
</form>