# Day 5: Handling Forms & Controlled Components in React

Today, you'll learn how to handle **forms** in React using **controlled components** and manage form data efficiently.

---

## 1 Forms in React

Forms allow users to input and submit data. In React, form elements like `<input>`, `<textarea>`, and `<select>` are **controlled components**, meaning their values are controlled by React state.

---

## 2 Controlled vs Uncontrolled Components

- **Controlled Component:** React controls the value of the input field via `useState()`.
- **Uncontrolled Component:** Uses **DOM directly** (`ref`) instead of React state (not recommended).

✅ **We use Controlled Components for better state management.**

---

## 3 Handling Input Fields with `useState()`

To capture user input, store it in state using `useState()`.

### Example: Simple Form Handling

```jsx
import { useState } from "react";

function SimpleForm() {
  const [name, setName] = useState("");

  return (
    <div>
      <input
        type="text"
        placeholder="Enter your name"
        value={name}
        onChange={(e) => setName(e.target.value)}
      />
      <p>Your name: {name}</p>
    </div>
  );
}

export default SimpleForm;
```

✅ **Whenever the user types, `setName()` updates the state, re-rendering the component.**

---

## 4️⃣ Handling Form Submission

When a form is submitted, React prevents the default browser behavior using `event.preventDefault()`.

### Example: Submitting a Form

```
import { useState } from "react";

function FormSubmit() {
  const [name, setName] = useState("");

  function handleSubmit(e) {
    e.preventDefault(); // Prevents page reload
    alert(`Submitted Name: ${name}`);
  }

  return (
    <form onSubmit={handleSubmit}>
      <input
        type="text"
        placeholder="Enter name"
        value={name}
        onChange={(e) => setName(e.target.value)}
      />
      <button type="submit">Submit</button>
    </form>
  );
}

export default FormSubmit;
```

✅ **This prevents full-page reload and handles the data in React state.**

---

## 5️⃣ Handling Multiple Form Inputs

For forms with multiple fields, use an **object state** to store all values.

### Example: Form with Multiple Inputs

```
import { useState } from "react";

function MultiInputForm() {
  const [formData, setFormData] = useState({ name: "", email: "" });

  function handleChange(e) {
    setFormData({ ...formData, [e.target.name]: e.target.value });
  }

  function handleSubmit(e) {
    e.preventDefault();
    alert(`Name: ${formData.name}, Email: ${formData.email}`);
  }

  return (
```

```
    <form onSubmit={handleSubmit}>
      <input
        type="text"
        name="name"
        placeholder="Name"
        value={formData.name}
        onChange={handleChange}
      />
      <input
        type="email"
        name="email"
        placeholder="Email"
        value={formData.email}
        onChange={handleChange}
      />
      <button type="submit">Submit</button>
    </form>
  );
}

export default MultiInputForm;
```

✅ **Dynamic updates using `[e.target.name]` ensure the correct field updates in state.**

---

## 6 Handling Select Dropdowns

React handles dropdowns just like text inputs.

### Example: Select Dropdown

```
import { useState } from "react";

function DropdownForm() {
  const [country, setCountry] = useState("India");

  return (
    <form>
      <select value={country} onChange={(e) => setCountry(e.target.value)}>
        <option value="India">India</option>
        <option value="USA">USA</option>
        <option value="UK">UK</option>
      </select>
      <p>Selected Country: {country}</p>
    </form>
  );
}

export default DropdownForm;
```

✅ **The selected value is stored in state and updates dynamically.**

---

## 7️⃣ Handling Checkboxes & Radio Buttons

Checkboxes & radio buttons work similarly, but for checkboxes, we use `checked` instead of `value`.

### Example: Checkbox Handling

```jsx
import { useState } from "react";

function CheckboxForm() {
  const [isSubscribed, setIsSubscribed] = useState(false);

  return (
    <form>
      <label>
        <input
          type="checkbox"
          checked={isSubscribed}
          onChange={() => setIsSubscribed(!isSubscribed)}
        />
        Subscribe to newsletter
      </label>
      <p>{isSubscribed ? "Subscribed" : "Not Subscribed"}</p>
    </form>
  );
}

export default CheckboxForm;
```

✅ **Checkbox state updates on click.**

---

## 8️⃣ Handling Radio Buttons

For radio buttons, ensure only one option is selected at a time.

### Example: Radio Button Handling

```jsx
import { useState } from "react";

function RadioForm() {
  const [gender, setGender] = useState("male");

  return (
    <form>
      <label>
        <input
          type="radio"
          name="gender"
          value="male"
          checked={gender === "male"}
          onChange={(e) => setGender(e.target.value)}
        />
        Male
      </label>
      <label>
        <input
```

```
          type="radio"
          name="gender"
          value="female"
          checked={gender === "female"}
          onChange={(e) => setGender(e.target.value)}
        />
        Female
      </label>
      <p>Selected Gender: {gender}</p>
    </form>
  );
}

export default RadioForm;
```

✅ **The selected radio button updates `gender` state.**

---

## 📌 Summary of Day 5

- ✔ **Controlled Components** keep form input data in React state
- ✔ **Handle form submission** with `event.preventDefault()`
- ✔ **Manage multiple inputs** dynamically
- ✔ **Work with dropdowns, checkboxes, and radio buttons**
- ✔ **Built interactive forms with React!**

---

## 🚀 Next Step: Day 6 - React Hooks (useEffect & useRef)