Day 8: React Forms & Controlled Components

Today, you'll learn how to handle **forms in React**, manage user input, and work with **controlled components** using `useState`.

---

1 What is a Controlled Component?

In React, a **controlled component** is a form element (input, textarea, select) whose **value is controlled by React state**.

- ✅ Form data is managed by React state
- ✅ UI updates automatically when state changes
- ✅ Easy validation & form submission handling

---

2 Handling User Input with `useState`

Example: Simple Input Field

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

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

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

export default SimpleForm;
```

- ✅ **`value={name}` makes it controlled**
- ✅ **Updating `setName(e.target.value)` keeps state in sync**

---

3 Handling Form Submission

Example: Submitting a Form

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

function FormSubmit() {
```

```
  const [email, setEmail] = useState("");

  const handleSubmit = (e) => {
    e.preventDefault();
    alert(`Submitted Email: ${email}`);
  };

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

export default FormSubmit;
```

✅ **Prevents page reload (`e.preventDefault()`)**
✅ **Captures form input on submit**

Handling Multiple Inputs with a Single State

Example: Managing Multiple Fields

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

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

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

  const handleSubmit = (e) => {
    e.preventDefault();
    console.log("Form Data:", formData);
  };

  return (
    <form onSubmit={handleSubmit}>
      <input
        name="name"
        value={formData.name}
        onChange={handleChange}
        placeholder="Name"
      />
      <input
        name="email"
        value={formData.email}
```

```
        onChange={handleChange}
        placeholder="Email"
      />
      <button type="submit">Submit</button>
    </form>
  );
}

export default MultiInputForm;
```

✅ **Handles multiple fields with one `useState`**
✅ **Uses `name` attribute to update correct field**

5️⃣ Handling Checkboxes, Radio Buttons, and Select Dropdowns

Example: Checkbox Input

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

function CheckboxForm() {
  const [isChecked, setIsChecked] = useState(false);

  return (
    <div>
      <input
        type="checkbox"
        checked={isChecked}
        onChange={(e) => setIsChecked(e.target.checked)}
      />
      <p>{isChecked ? "Checked" : "Unchecked"}</p>
    </div>
  );
}

export default CheckboxForm;
```

✅ **`checked` is controlled by state**

Example: Radio Buttons

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

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

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

```
        />
        Male
      </label>
      <label>
        <input
          type="radio"
          value="Female"
          checked={gender === "Female"}
          onChange={(e) => setGender(e.target.value)}
        />
        Female
      </label>
      <p>Selected: {gender}</p>
    </div>
  );
}

export default RadioForm;
```

✅ **Radio buttons are controlled by `useState`**

Example: Select Dropdown

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

function SelectForm() {
  const [color, setColor] = useState("");

  return (
    <div>
      <select value={color} onChange={(e) => setColor(e.target.value)}>
        <option value="">Choose a color</option>
        <option value="Red">Red</option>
        <option value="Blue">Blue</option>
        <option value="Green">Green</option>
      </select>
      <p>Selected Color: {color}</p>
    </div>
  );
}

export default SelectForm;
```

✅ **Handles dropdown selection with `useState`**

---

6 Validating Forms Before Submission

You can validate inputs **before allowing submission**.

Example: Simple Validation

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

```jsx
function ValidatedForm() {
  const [email, setEmail] = useState("");
  const [error, setError] = useState("");

  const handleSubmit = (e) => {
    e.preventDefault();
    if (!email.includes("@")) {
      setError("Invalid email format");
      return;
    }
    alert(`Submitted Email: ${email}`);
  };

  return (
    <form onSubmit={handleSubmit}>
      <input
        type="email"
        value={email}
        onChange={(e) => setEmail(e.target.value)}
        placeholder="Enter email"
      />
      {error && <p style={{ color: "red" }}>{error}</p>}
      <button type="submit">Submit</button>
    </form>
  );
}

export default ValidatedForm;
```

✅ **Checks if email contains @ before submission**

📌 Summary of Day 8

- ✔ **Controlled components** use `value` & `onChange`
- ✔ **Form submission** is handled with `onSubmit`
- ✔ **Multiple inputs** can be managed using a single state object
- ✔ **Checkbox, radio buttons, and select** are controlled using `useState`
- ✔ **Form validation** ensures correct data before submission

---

🚀 Next Step: Day 9 - React Context API (State Management)