Module 4 — React + Vite: Forms & State with `useState` (80 minutes)

> Audience: Kids 13–14 (continuing from Module 3). We keep building in the same `react-app` project. Today's code lives in `src/day4/`.

Learning Objectives

By the end of this module, students will: 1. Understand what **state** is and why React needs it. 2. Use the `useState` **hook** to store and update data. 3. Build **controlled inputs** (React controls the value of the input). 4. Handle **form submission** safely in React. 5. Add **basic validation** and show messages to the user.

80-Minute Agenda

- **0–10 min** — Recap: events, lists, Todo app
- **10–25 min** — What is state? Why do we need it?
- **25–40 min** — Controlled inputs (`value` + onChange)
- **40–60 min** — Form submission + validation
- **60–75 min** — Hands-on mini-project: Add Items Form
- **75–80 min** — Recap, Q&A, Homework

Recap (0–10 min)

Ask students: - What happens when we click a button in React? - Why did we use `useState` in the Counter/Todo apps? - What does `map()` help us do?

What Is State? (10–25 min)

Simple Explanation

- **State = memory of a component**.
- If data changes over time and should update the UI → it belongs in **state**.

Real-Life Analogy

- A scoreboard remembers the score.
- When a goal is scored, the number changes.
- In React, **state** is the scoreboard.

Why Not Normal Variables?

```
let count = 0; // ❌ React will NOT re-render when this changes
```

- React does not watch normal variables. - React **does** watch state.

`useState` Basics

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

const [value, setValue] = useState(0);
```

- `value` → current state value - `setValue` → function to update it - `0` → initial value

---

Controlled Inputs (25–40 min)

What Is a Controlled Input?

- React controls the input's value.
- The input's value comes from **state**.

Example: Name Input

Create `src/day4/NameInput.jsx`:

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

function NameInput() {
  const [name, setName] = useState(''); // state for input text

  function handleChange(event) {
    setName(event.target.value); // update state when typing
  }

  return (
    <div className="card">
      <input
        type="text"
        value={name}              // input value comes from state
        onChange={handleChange}   // update state on every keystroke
        placeholder="Enter your name"
      />
```

```
      <p>Hello, {name}</p> {/* UI updates automatically */}
    </div>
  );
}

export default NameInput;
```

Teaching Points

- `event.target.value` = what the user typed.
- One source of truth: **state**.

Forms in React (40–60 min)

Prevent Default Behavior

HTML forms reload the page by default — we must stop that.

Example: Simple Form

Create `src/day4/SimpleForm.jsx`:

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

function SimpleForm() {
  const [email, setEmail] = useState('');
  const [error, setError] = useState('');

  function handleSubmit(event) {
    event.preventDefault(); // stop page reload

    if (email.trim() === '') {
      setError('Email is required'); // basic validation
      return;
    }

    setError('');
    alert('Submitted: ' + email);
    setEmail(''); // reset input
  }

  return (
    <form className="card" onSubmit={handleSubmit}>
      <h2>Subscribe</h2>
```

```
      <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 SimpleForm;
```

Key Concepts

- `onSubmit` handles the form.
- `event.preventDefault()` stops refresh.
- Conditional rendering: `{error && <p>}`

Hands-On Mini-Project (60–75 min)

Goal

Build a form that adds items to a list.

Create `src/day4/ItemForm.jsx`:

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

function ItemForm() {
  const [items, setItems] = useState([]); // list of items
  const [text, setText] = useState('');

  function addItem(event) {
    event.preventDefault();

    if (text.trim() === '') return; // ignore empty

    setItems([...items, text]); // add item to list
    setText(''); // clear input
```

```
  }

  return (
    <div className="card">
      <form onSubmit={addItem}>
        <input
          value={text}
          onChange={e => setText(e.target.value)}
          placeholder="Add item"
        />
        <button>Add</button>
      </form>

      <ul>
        {items.map((item, index) => (
          <li key={index}>{item}</li>
        ))}
      </ul>
    </div>
  );
}

export default ItemForm;
```

Use in `App.jsx`:

```
import NameInput from './day4/NameInput.jsx';
import SimpleForm from './day4/SimpleForm.jsx';
import ItemForm from './day4/ItemForm.jsx';

<main className="container">
  <NameInput />
  <SimpleForm />
  <ItemForm />
</main>
```

Common Mistakes (Explain Slowly)

- ❌ Forgetting `event.preventDefault()` → page refreshes
- ❌ Using normal variables instead of state
- ❌ Not binding `value` to input
- ❌ Mutating state directly (`items.push()`)

Recap (75–80 min)

- State stores data that changes.
- `useState` lets React track changes.
- Controlled inputs = value comes from state.
- Forms need `onSubmit` + `preventDefault()`.

Homework

1. Create a `RegisterForm.jsx` with:
2. Username
3. Password
4. Age
5. Validate:
6. No empty fields
7. Age must be ≥ 10
8. Show error messages in red.

Bonus: Clear the form after successful submit.

Instructor Script (Suggested)

- “If data changes and affects the screen, it must be state.”
- “React controls the input — not the browser.”
- “Forms in React never reload the page.”
- “Validation is just logic before updating state.”