Handle Input Field

In React, unlike plain HTML, we don’t directly access the DOM using document.getElementById.

Instead, React controls the input’s value using state — this makes it a Controlled Component.

This allows React to:

- Keep UI and data in sync
- Validate or modify input instantly
- Easily clear/reset form values
- Make complex forms predictable

```jsx
const [name, setName] = useState("");

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

value → connected to state

onChange → updates state

Without onChange, the input becomes read-only

What is Controlled Components?

A controlled component is a form whose input field value is controlled by React's State.

Here's how it works:

- Store input field value in state
- Use change handler with input field
- Value attribute attached with state

Uncontrolled Components

Sometimes we don’t bind the input to state — we directly access it via ref.

```
const inputRef = useRef();

function handleSubmit() {
  alert("Input value: " + inputRef.current.value);
}

return (
  <div>
    <input ref={inputRef} type="text" placeholder="Type here..." />
    <button onClick={handleSubmit}>Submit</button>
  </div>
);
```

Multiple Input Fields

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

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

return (
  <form>
    <input
      name="name"
      value={formData.name}
      onChange={handleChange}
      placeholder="Name"
    />
    <input
      name="email"
      value={formData.email}
      onChange={handleChange}
      placeholder="Email"
    />
    <h3>{formData.name} -- {formData.email}</h3>
  </form>
);
```

When handling multiple inputs, use one state object.

Dynamic key [name]: value

Spread operator ...prev to keep previous data intact

Thank You