🧠 Controlled vs Uncontrolled Components in React

In React, form elements like `<input>`, `<textarea>`, and `<select>` can work in **two ways**:

✅ 1. **Controlled Component**

In a **controlled component**, the form element's value is **controlled by React state**.

Example:

jsx Copy Edit

```jsx
import { useState } from 'react'; function ControlledInput() { const [name, setName] = useState(''); return ( <div> <input type="text" value={name} // Controlled by state onChange={(e) => setName(e.target.value)} /> <p>You typed: {name}</p> </div> ); }
```

🔍 Key Points:

- State is **the single source of truth**.
- You **control** the value using `useState`.
- Good for validations, live preview, etc.

❌ 2. **Uncontrolled Component**

In an **uncontrolled component**, the form element **manages its own state**. You **access** the value using a `ref`.

Example:

jsx Copy Edit

```jsx
import { useRef } from 'react'; function UncontrolledInput() { const inputRef = useRef(); const handleSubmit = () => { alert(`Input value: ${inputRef.current.value}`); }; return ( <div> <input type="text" ref={inputRef} /> <button onClick={handleSubmit}>Submit</button> </div> ); }
```

🔍 Key Points:

- Value is handled by the **DOM**.
- React doesn't manage or track it.
- Simpler, but **less powerful** than controlled.

⚖️ Controlled vs Uncontrolled Summary

Feature	Controlled	Uncontrolled
Value managed by	React state	DOM (via `ref`)
Easier to validate	✅ Yes	❌ Harder
Two-way binding	✅ Yes	❌ No
Simpler for quick use	❌ Slightly more setup	✅ Yes

🧪 When to Use What?

- **Controlled**: Forms with validations, conditional rendering, or live preview
- **Uncontrolled**: Quick forms, file uploads, or when integrating with non-React libraries