React Inputs and Forms

Introduction

In React, users can provide input through several different UI elements. These are the common **HTML form controls** that React can manage:

Input Type	Element	Use Case
Text	`<input type="text">`	Basic single-line input
Password	`<input type="password">`	Hidden text input
Textarea	`<textarea>`	Multiline input
Checkbox	`<input type="checkbox">`	True/false input
Radio	`<input type="radio">`	Single choice from group
Select	`<select>`	Drop-down list
File	`<input type="file">`	Uploading files
Range	`<input type="range">`	Slider for number input

React supports **controlled** and **uncontrolled** components for all of the above.

Uncontrolled Inputs

In **uncontrolled inputs**, React doesn't manage the input's state — **the DOM does**. You use a **ref** to access the value when you need it

Example: Using onKeyUp with an Uncontrolled Input

In below Example

- ref={inputRef} keeps a reference to the DOM input.
- onKeyUp={handleKeyUp} fires every time a key is released.
- Inside handleKeyUp, you can:
    - Access the key that was pressed (e.key)
    - Read the input value using inputRef.current.value

```jsx
import React, { useRef } from 'react';

function KeyUpExample() {
  const inputRef = useRef();

  const handleKeyUp = (e) => {
    console.log('Key pressed:', e.key);
    console.log('Current value:', inputRef.current.value);
  };

  return (
    <div>
      <h2>Type something</h2>
      <input type="text" ref={inputRef} onKeyUp={handleKeyUp} />
    </div>
  );
}

export default KeyUpExample;
```

useRef is a React Hook that lets you persist values across renders without causing re-renders, and it can also be used to access DOM elements directly**.**

Note : We will see more about useRef in Hooks Chapter.

Controlled Inputs

A **controlled input** in React is an input element (like <input>, <textarea>, <select>, etc.) whose value is controlled by React state**.**
React becomes the **"single source of truth"** for that input's value.

Example of a Controlled Input

```jsx
import React, { useState } from 'react';

function ControlledInput() {
  const [name, setName] = useState('');

  const handleChange = (e) => {
    setName(e.target.value);
  };

  return (
    <div>
      <input type="text" value={name} onChange={handleChange} />
      <p>Hello, {name}!</p>
    </div>
  );
}
```

Key elements:

- value={name} → React **controls** the input's value.
- onChange updates the state whenever the user types.
- The UI (input field) always reflects the name state.

Why Use Controlled Inputs?

Pros:

- Easy to **validate** or **transform** input (e.g. uppercase, numbers only).
- Keeps form data in sync with state.
- Enables **instant feedback**, like live previews.
- Makes it easy to **reset forms** or trigger effects based on input.

Cons:

- Slightly more verbose (more code for simple things).
- Can feel like overkill for very simple forms.
- Typing might feel sluggish (laggy) in very large forms without optimization.