Forms in React

Question 1: How do you handle forms in React? Explain the concept of controlled components.

Ans.

In React, forms are handled by connecting form inputs to component state. Instead of the browser managing input values (like in plain HTML), React manages the data.

There are two main ways:

1. Controlled Components (recommended & most common)
2. Uncontrolled Components (less common)

What is a Controlled Component?

A controlled component is a form element whose value is controlled by React state.

Single source of truth = React state

Key idea:

- Input value comes from useState
- Changes are handled using onChange
- React decides what the input shows

Controlled Component Flow (Very Important)

1. User types in input
2. onChange event fires
3. State gets updated using setState

4. Input value updates from state

Example: Controlled Input

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

function LoginForm() {
  const [email, setEmail] = useState("");

  return (
    <form>
      <input
        type="email"
        value={email}
        onChange={(e) => setEmail(e.target.value)}
        placeholder="Enter email"
      />

      <p>Email: {email}</p>
    </form>
  );
}

export default LoginForm;
```

Why this is controlled :

value is coming from state

Input cannot change without setEmail

React fully controls the input

Handling Multiple Inputs :

```
const [formData, setFormData] = useState({
  username: "",
  password: ""
});

function handleChange(e) {
  const { name, value } = e.target;
  setFormData({ ...formData, [name]: value });
}
<input
  name="username"
  value={formData.username}
  onChange={handleChange}
/>

<input
```

```
  name="password"
  value={formData.password}
  onChange={handleChange}
/>
```

Handling Form Submission :

```
<form onSubmit={handleSubmit}>
```

<u>Why Use Controlled Components :</u>

Advantages

- Easy validation
- Real-time form updates
- Better control over user input
- Easy to reset form
- Works well with libraries like Formik, React Hook Form

Disadvantage

- Slightly more code compared to plain HTML

Controlled vs Uncontrolled (Quick Comparison)

Feature	Controlled	Uncontrolled
Data handled by	React State	DOM
Uses useState	Yes	No
Validation	Easy	Hard
Recommended	Yes	No

Question 2: What is the difference between controlled and uncontrolled components in React?

Ans .

<u>**Controlled Components**</u>

A controlled component is a form element whose value is controlled by React state.

Key points:

- Uses useState
- Value comes from state
- Updated using onChange
- React is the single source of truth

Example (Controlled Input)

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

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

  return (
    <input
      type="text"
      value={name}
      onChange={(e) => setName(e.target.value)}
    />
  );
}
```

Input value = React state

Fully controlled by React

Uncontrolled Components

An uncontrolled component stores form data directly in the DOM, not in React state.

Key points:

- Uses useRef
- No useState
- React does not control the input value

- DOM is the source of truth

Example (Uncontrolled Input)

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

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

 function handleSubmit() {
  console.log(inputRef.current.value);
 }

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

Value accessed from DOM

React doesn’t manage input value

Controlled vs Uncontrolled (Comparison Table)

Feature	**Controlled**	**Uncontrolled**
Data source	React State	DOM
Uses useState	Yes	No
Uses useRef	No	Yes
Real-time validation	Easy	Hard
Form control	Full	Limited
Performance	Slightly slower	Slightly faster
Recommended	Yes	Rare

When to Use What :

Use Controlled Components when:

- You need validation
- You want real-time updates
- You need dynamic behavior
- You want predictable state

Use Uncontrolled Components when:

- Simple forms
- Quick prototyping
- Integrating with non-React libraries
