FormExample.js :-

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

function FormExample() {
  const [formData, setFormData] = useState({
    name: "",
    email: "",
    password: ""
  });

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

    setFormData({
      ...formData,
      [name]: value, // dynamic key update
    });
  };

  const handleSubmit = (e) => {
    e.preventDefault();
    alert(JSON.stringify(formData, null, 2));
  };
```

```
return (
   <form onSubmit={handleSubmit}>
     <h2>React Form</h2>

     <div>
      <label>Name: </label>
      <input
        type="text"
        name="name"
        value={formData.name}
        onChange={handleChange}
      />
     </div>

     <div>
      <label>Email: </label>
      <input
        type="email"
        name="email"
        value={formData.email}
        onChange={handleChange}
      />
     </div>

     <div>
```

```
        <label>Password: </label>
        <input
          type="password"
          name="password"
          value={formData.password}
          onChange={handleChange}
        />
      </div>

      <button type="submit">Submit</button>
    </form>
{/*  Preview Section */}
      <div style={{ marginTop: "20px" }}>
        <h3>Preview Data</h3>
        <p><strong>Name:</strong> {formData.name}</p>
        <p><strong>Email:</strong> {formData.email}</p>
        <p><strong>Password:</strong> {formData.password}</p>
      </div>
    </div>
  );
}

export default FormExample;
```

□ Line-by-Line Explanation

---

1. Importing React Hook

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

- Imports the **useState** hook from React.
- useState lets your component store and update values (state).

---

2. Component Declaration

```
function FormExample() {
```

- Creates a functional component named **FormExample**.
- Everything inside this function is the component's code.

---

3. Declaring State for the Form

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

Breakdown:

- `formData` → holds all input values in an object.
- `setFormData` → function to update the `formData`.
- `useState({...})` → initializes the state with an object:
    - `name: ""` → empty name
    - `email: ""` → empty email
    - `password: ""` → empty password

So initially the form fields are empty.

---

4. Handling Input Changes

```
const handleChange = (e) => {
```

- Creates a function that runs when the user types in an input.

```
const { name, value } = e.target;
```

- Extracts `name` and `value` from the input that triggered the event.
- Example: if user types in email field
    - `name = "email"`
    - `value = "hello@example.com"`

```
setFormData({
  ...formData,
  [name]: value,
});
```

Breakdown:

- `...formData` → copies the old formData.
- `[name]: value` → updates only the field that changed (dynamic key).
    - if `name = "email"`, this becomes → `email: value`

This keeps other fields unchanged while updating one.

---

5. Handling Form Submission

```
const handleSubmit = (e) => {
```

- A function that runs when user submits the form.

```
e.preventDefault();
```

- Stops the page from refreshing (default browser behavior).
- Allows React to handle the submission.

```
alert(JSON.stringify(formData, null, 2));
```

- Turns the formData object into a readable string.
- Shows it in an alert popup.
- `null, 2` makes the JSON nicely formatted.

---

6. JSX Returned by the Component

```
return (
```

- Everything inside return () is what the component displays on the screen.

---

7. Form Element

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

- Creates a form.
- When the user submits it, the `handleSubmit` function runs.

---

8. Heading

```
<h2>React Form</h2>
```

- Displays a title.

---

9. Name Input

```
<div>
  <label>Name: </label>
  <input
    type="text"
    name="name"
    value={formData.name}
    onChange={handleChange}
  />
</div>
```

Explanation:

- `<input />` is a text input.
- `name="name"` → this matches the state key.
- `value={formData.name}` → input value comes from state.
- `onChange={handleChange}` → updates state when typing.

---

10. Email Input

Same idea:

```
<input
  type="email"
```

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

---

11. Password Input

Same idea:

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

---

12. Submit Button

```
<button type="submit">Submit</button>
```

- When clicked, the form submits.
- React calls `handleSubmit`.

---

13. Close JSX and Component

```
</form>
  );
}

export default FormExample;
```

- Ends the form.
- Ends the component.
- `export default` makes it available to use in other files.

□ Summary (Very Short)

Code	Meaning
useState	Stores form values
formData	Object with name, email, password
handleChange	Updates specific field dynamically
handleSubmit	Prevents reload + shows form data
value={}	Makes input controlled
onChange={}	Updates state while typing