5. Forms and Validation in React

Last updated by | Subramanya Dixit | May 5, 2025 at 7:59 PM GMT+5:30

📘 Why Forms and Validation Are Important

Forms are essential for gathering user input, and proper validation ensures data integrity. React provides powerful ways to manage forms using state and controlled components.

🧠 Key Concepts

✅ Controlled Components

Form inputs whose value is driven by React state.

```
function NameForm() {
  const [name, setName] = useState('');
  return (
    <input value={name} onChange={(e) => setName(e.target.value)} />
  );
}
```

✅ Handling Form Submission

Prevent default browser behavior and use `onSubmit` handler.

```
function Form() {
  const handleSubmit = (e) => {
    e.preventDefault();
    console.log('Form submitted');
  };
  return <form onSubmit={handleSubmit}>...</form>;
}
```

✅ Basic Validation

Perform input checks before submission.

```
function FormWithValidation() {
  const [email, setEmail] = useState('');
  const [error, setError] = useState('');

  const handleSubmit = (e) => {
    e.preventDefault();
    if (!email.includes('@')) {
      setError('Invalid email');
    } else {
      setError('');
      console.log('Submitted:', email);
    }
  };

  return (
    <form onSubmit={handleSubmit}>
      <input value={email} onChange={(e) => setEmail(e.target.value)} />
      {error && <p>{error}</p>}
      <button type="submit">Submit</button>
    </form>
  );
}
```

✅ Form Libraries (Formik or React Hook Form)

Simplify form logic with third-party libraries.

```
// With React Hook Form
import { useForm } from 'react-hook-form';

function RHFForm() {
  const { register, handleSubmit } = useForm();
  const onSubmit = (data) => console.log(data);

  return (
    <form onSubmit={handleSubmit(onSubmit)}>
      <input {...register("username")} />
      <button type="submit">Submit</button>
    </form>
  );
}
```

📊 Visual Overview

```
flowchart TD
  A[User Input] --> B[Controlled Component]
  B --> C[Validation Logic]
  C --> D[Submit Handler]
  D --> E[Processed Data]
```

💡 Guidelines

- Use controlled components for form input.
- Validate on submit or as user types.
- Display clear feedback for errors.
- Prefer libraries like React Hook Form for complex forms.

📝 Practice Exercises

1. Create a basic form with name and email inputs.
2. Add simple validation to check if both fields are filled.
3. Use `useState` to manage error messages.
4. Rebuild the form using `react-hook-form`.

❓ Quiz Questions

1. What is a controlled component?
a) A component with local storage
b) An uncontrolled HTML form
✅ **c) A form input whose value is driven by React state**
d) A styled component

2. What does `e.preventDefault()` do in a form submission?
a) Validates the input
✅ **b) Prevents page reload**
c) Adds default props
d) Resets form state

3. Why use React Hook Form or Formik?
a) For API integration
b) To reduce bundle size
✅ **c) To simplify form handling and validation**
d) For theming support

4. When should you show error messages in a form?
a) Only after submission
b) Never, use browser validation
c) Only on mobile
✅ **d) As user types or on submit**