🎯 Working with **Events** in React

React makes handling events **easy and consistent** across browsers. Events in React are very similar to DOM events, but they use a **camelCase syntax** and pass a **SyntheticEvent** instead of a real DOM event.

🧠 Key Concepts

- Events in React are written in **camelCase** (e.g., `onClick` instead of `onclick`).
- You pass a **function**, not a string.
- React wraps native browser events in a **SyntheticEvent** for performance and cross-browser support.

📦 Example: Click Event

jsx Copy Edit

```jsx
function ClickButton() { function handleClick() { alert("Button clicked!"); } return <button
onClick={handleClick}>Click Me</button>; }
```

🔤 Example: Input Event (onChange)

jsx Copy Edit

```jsx
import { useState } from 'react'; function InputBox() { const [text, setText] = useState(''); const
handleChange = (event) => { setText(event.target.value); }; return ( <div> <input type="text"
onChange={handleChange} /> <p>You typed: {text}</p> </div> ); }
```

✅ Common React Events

Event	Description
`onClick`	When a button or element is clicked
`onChange`	For inputs, textareas, selects
`onSubmit`	When a form is submitted
`onMouseEnter`	When mouse enters an element
`onMouseLeave`	When mouse leaves an element
`onKeyDown`	When a key is pressed
`onFocus`	When an element gains focus
`onBlur`	When an element loses focus

🧹 Inline Event Handler

You can also define inline like this:

jsx Copy Edit

```jsx
<button onClick={() => alert('Clicked!')}>Click</button>
```

🧠 Tip: Prevent Default Behavior

jsx Copy Edit

```jsx
function MyForm() { const handleSubmit = (e) => { e.preventDefault(); // prevents form from reloading the page alert('Form submitted!'); }; return ( <form onSubmit={handleSubmit}> <button type="submit">Submit</button> </form> ); }
```