Handling Events in React

In this lecture, we will learn how to handle user interactions in React using events. Events make applications interactive, such as responding to clicks, typing, or form submissions.

Event Handling in React

React events are very similar to DOM events in plain JavaScript, but there are a few key differences:

- Event names are written in **camelCase** (e.g., `onClick` instead of `onclick`).
- You pass a function as the event handler, not a string.

Example: Button Click

```
function App() {
  function handleClick() {
    alert("Button was clicked!")
  }

  return (
    <div>
      <button onClick={handleClick}>Click Me</button>
    </div>
  )
}

export default App
```

Here, when the button is clicked, the `handleClick` function runs.

Inline Event Handlers

You can also define the function inline:

```
<button onClick={() => alert("Inline click!")}>Click Inline</button>
```

While this works, using separate functions is better for readability when the logic gets more complex.

Example: Updating State with Events

Let's combine events with state:

```
import { useState } from "react"

function App() {
  const [text, setText] = useState("Hello")

  function updateText() {
    setText("You clicked the button!")
  }

  return (
    <div>
      <p>{text}</p>
      <button onClick={updateText}>Change Text</button>
    </div>
  )
}

export default App
```

Clicking the button updates the state, and React re-renders the UI.

Recap

- React event handlers use camelCase names like `onClick` and `onChange`.
- Event handlers are functions that run in response to user actions.
- Combining events with state allows us to create dynamic, interactive components.