**Hands-on: 11. ReactJS-HOL**

**Introduction**

In React, user interactions such as clicks, typing, or form submissions are managed using events. React provides a consistent way to handle these events across all browsers using a concept called Synthetic Events. Event handling in React is declarative, clean, and follows a specific naming convention. This section explores how React handles events, what synthetic events are, and how to use event handlers effectively.

1. **React Events**

React events are the way to handle user interactions in a React application. These events are very similar to DOM events but have some differences and improvements:

* Example:

<button onClick={handleClick}>Click Me</button>

React supports all common browser events such as:

* Mouse events: onClick, onMouseOver, onMouseDown
* Form events: onChange, onSubmit
* Keyboard events: onKeyDown, onKeyUp
* Focus events: onFocus, onBlur

1. **Event Handlers**

Event handlers are functions that execute when an event occurs. In React, these handlers are usually defined as methods or arrow functions.

* Example:

function handleClick() {

alert("Button clicked!");

}

<button onClick={handleClick}>Click</button>

* Event handlers can also receive the event object:

function handleClick(event) {

console.log(event.target);

}

* Key points:
* Event handlers are passed as references, not as a function call (onClick={handleClick} not onClick={handleClick()})
* You can bind handlers using arrow functions or .bind() to maintain this context.

1. **Synthetic Events**

React uses a wrapper around the browser’s native events called a Synthetic Event. It’s part of React’s event delegation system, providing cross-browser consistency and better performance.

* Features of Synthetic Events:
* Wraps the native event with a unified API
* Works identically across all browsers
* Automatically pooled for performance (in earlier versions)
* Example:

function handleInput(event) {

console.log(event.type); // 'change'

}

<input type="text" onChange={handleInput} />

Synthetic events have all the same methods and properties as native events (e.g., preventDefault(), stopPropagation()).

1. **React Event Naming Convention**

React follows camelCase naming for event attributes instead of lowercase (as in plain HTML):

|  |  |
| --- | --- |
| HTML | React |
| onclick | onClick |
| onchange | onChange |
| onsubmit | onSubmit |
| onmouseover | onMouseOver |

Also:

* The handler value must be a function, not a string.
* JSX event attributes are written directly on the element tag.
* Example:

<button onclick="alert('Hi')">Wrong</button> ❌

<button onClick={() => alert('Hi')}>Correct</button> ✅

**Conclusion**

React streamlines event handling through Synthetic Events and a consistent API across browsers. By using camelCase event names and function-based handlers, React encourages clean and organized interaction logic. Mastery of React events and event handlers is essential for building responsive, interactive web applications.