**WEEK 7: REACT HO 11**

**1. Concepts & Definitions**

**React Events**

React events are wrappers around the browser’s native events. They work almost the same as DOM events but have some syntactic differences and benefits (like cross-browser normalization).

**Event Handlers**

An event handler is a function you pass to a React element that should run when a particular event occurs. For example:

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

handleClick is the event handler.

**2. Synthetic Event**

React creates a cross-browser wrapper called a **SyntheticEvent** around the native event. It normalizes behavior so your code works consistent across browsers. You get it as the first argument in handlers, e.g.:

const handleClick = (event) => {

console.log(event.type); // SyntheticEvent

};

Note: Synthetic events are pooled for performance; if you want to use the event asynchronously, call event.persist().

**3. Naming Convention**

React uses **camelCase** for event prop names, e.g.:

* Correct: onClick, onChange, onSubmit
* Incorrect: onclick, OnClick, "handleClick()" (you pass a function reference, not a string or call)

Also: pass the function, not its invocation unless you wrap it:

// Good

<button onClick={doSomething} />

// If you need to pass args:

<button onClick={() => doSomething("welcome")} />

**this keyword**

* In **functional components** (with hooks) you don’t use this.
* In **class components**, if you reference this inside a handler, you must bind methods in constructor or use arrow class fields:
* // binding in constructor
* this.handleClick = this.handleClick.bind(this);

For this lab, we’ll use **functional components with hooks**—clean and modern, no need to deal with this.

Anshika Srivastava  
superset id: 6387029