1. Explain React Events

React **events** are similar to DOM events in HTML (like onclick, onchange, etc.), but with a few differences:

- React wraps native events inside a **Synthetic Event** system for better cross-browser compatibility.
- Events in React are handled using **camelCase syntax** (e.g., onClick, onChange).

Example:

jsx

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

2. Explain Event Handlers

An **event handler** in React is a **function that is called when a specific event occurs** (e.g., a button is clicked).

You define it like a regular function and pass it to an event prop like onClick, onChange, etc.

Example:

jsx

function handleClick() {

  alert("Button was clicked");

}

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

You can also use arrow functions:

jsx

<button onClick={() => alert("Clicked!")}>Click</button>

3. Define Synthetic Event

A **Synthetic Event** in React is a **wrapper around the browser's native event** that provides **consistent behavior across all browsers**.

React uses SyntheticEvent to normalize events so that you don't have to worry about browser differences.

It has the same interface as the browser's native event, including methods like preventDefault() and stopPropagation().

Example:

jsx

```
function handleClick(e) {
  e.preventDefault(); // prevents default behavior
  console.log("Synthetic event triggered");
}
```

4. React Event Naming Convention

React uses a consistent **camelCase naming convention** for events:

HTML Event	React Event
onclick	onClick
onchange	onChange
onmouseover	onMouseOver
onkeydown	onKeyDown
onsubmit	onSubmit

The value passed to these event props must be a **function reference**, not a string.