## ReactJS – Week 1, Day 6

### Topic: Event Handling – onClick, onChange, Handling User Input

### 🔹 What is Event Handling?

Event handling in React refers to the process of managing and responding to user interactions (like clicks, typing, form submissions, etc.) with our application.

* In plain HTML/JavaScript, we attach event listeners to elements.
* In React, we use built-in props like onClick, onChange, onSubmit, etc., to handle these events.

Event handling makes our application **interactive and dynamic**.

### 🔹 Why do we use Event Handling?

* To **respond to user actions** (e.g., button clicks, text input changes).
* To **update the state** of components when something happens.
* To **control user input** (forms, checkboxes, radio buttons, etc.).
* To build interactive UI elements like dropdowns, search bars, and calculators.

### 🔹 onClick

The onClick event handler is triggered when a user clicks on an element (usually a button).

**Example:**

import React, { useState } from "react";  
  
function ClickExample() {  
 const [count, setCount] = useState(0);  
  
 const handleClick = () => {  
 setCount(count + 1);  
 };  
  
 return (  
 <div>  
 <h2>Button clicked {count} times</h2>  
 <button onClick={handleClick}>Click Me</button>  
 </div>  
 );  
}  
  
export default ClickExample;

✅ When the button is clicked, the handleClick function runs and updates the state.

### 🔹 onChange

The onChange event handler is triggered when the value of an input element changes.

**Example:**

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

✅ Whatever the user types in the input box is reflected immediately in the UI.

### 🔹 Handling User Input

When handling user input, we typically: 1. Store the input value in **state** using useState. 2. Update the state whenever the user changes the input (onChange). 3. Use the state value wherever needed.

**Example: Simple Form**

import React, { useState } from "react";  
  
function FormExample() {  
 const [name, setName] = useState("");  
  
 const handleSubmit = (event) => {  
 event.preventDefault(); // Prevents page refresh  
 alert("Hello, " + name);  
 };  
  
 return (  
 <form onSubmit={handleSubmit}>  
 <input  
 type="text"  
 value={name}  
 onChange={(e) => setName(e.target.value)}  
 placeholder="Enter your name"  
 />  
 <button type="submit">Submit</button>  
 </form>  
 );  
}  
  
export default FormExample;

✅ This form takes user input and shows an alert when submitted.

### 🔹 Key Takeaways

* onClick → used for button clicks or any clickable elements.
* onChange → used for form inputs, checkboxes, dropdowns, etc.
* Handling user input requires **state** to store and update values.
* Events in React are written in **camelCase** (e.g., onClick not onclick).

### 📝 15–20 Minute Exercise

Build a **small login form** with the following requirements:

1. Create a component with two input fields: username and password.
2. Use useState to store values of both fields.
3. Add an onChange handler to update the state when the user types.
4. Add an onClick button called Login.
5. When clicked, display the entered username and password below the form.

**Bonus Task:** Add validation to check if both fields are not empty before showing the values.

✅ After completing this exercise, you’ll have a solid understanding of handling events and user input in React.