📌 Complete Example: Handling Click Events in React

Click events are **one of the most fundamental interactions** in a web application. In React, we handle click events using the `onClick` event listener, which allows us to execute functions when a user clicks on an element like a **button, link, or div**.

✅ Features in this Example

- ☑ **Using `onClick` in both Functional and Class Components**
- ☑ **Passing parameters in event handlers**
- ☑ **Preventing default behavior (e.g., links, forms)**
- ☑ **Using inline event handlers and separate functions**
- ☑ **Demonstrating event bubbling and propagation**

📝 `ButtonClick.js` (Functional Component with Click Event)

This component demonstrates handling click events **inside a functional component** using both **inline and separate functions**.

```
jsx
CopyEdit
import React, { useState } from "react";

const ButtonClick = () => {
  const [message, setMessage] = useState("Click the button!");

  // Function to handle button click
  const handleClick = () => {
    setMessage("Button clicked! 🥕");
  };

  // Function to handle click with parameters
  const handleClickWithParam = (name) => {
    setMessage(`Hello, ${name}! You clicked the button.`);
  };

  return (
    <div style={styles.container}>
```

```
      <h2>{message}</h2>
      {/* Click event with inline function */}
      <button style={styles.button} onClick={handleClick}>
        Click Me
      </button>

      {/* Click event with a function passing a parameter */}
      <button
        style={styles.button}
        onClick={() => handleClickWithParam("Manuel")}
      >
        Click with Name
      </button>
    </div>
  );
};

// Inline styles for simplicity
const styles = {
  container: {
    textAlign: "center",
    padding: "20px",
    border: "1px solid #ddd",
    borderRadius: "8px",
    width: "250px",
    margin: "20px auto",
    backgroundColor: "#f9f9f9",
  },
  button: {
    margin: "5px",
    padding: "10px",
    fontSize: "16px",
    cursor: "pointer",
  },
};

export default ButtonClick;
```

 **ClickEventClass.js** (Class Component with Click Event)

This example **shows how click events are handled inside class components** using this.setState and event binding.

```
jsx
CopyEdit
import React, { Component } from "react";

class ClickEventClass extends Component {
  constructor(props) {
    super(props);
```

```
    this.state = {
      message: "Click the button!",
    };
    this.handleClick = this.handleClick.bind(this); // Binding in constructor
  }

  // Method to update state on button click
  handleClick() {
    this.setState({ message: "Button clicked in Class Component! 🎉" });
  }

  render() {
    return (
      <div style={styles.container}>
        <h2>{this.state.message}</h2>
        <button style={styles.button} onClick={this.handleClick}>
          Click Me (Class Component)
        </button>
      </div>
    );
  }
}

const styles = {
  container: {
    textAlign: "center",
    padding: "20px",
    border: "1px solid #ddd",
    borderRadius: "8px",
    width: "250px",
    margin: "20px auto",
    backgroundColor: "#f9f9f9",
  },
  button: {
    margin: "5px",
    padding: "10px",
    fontSize: "16px",
    cursor: "pointer",
  },
};

export default ClickEventClass;
```

App.js (Parent Component to Display Both Examples)

This file renders both **functional and class components** that handle click events.

jsx
CopyEdit

```
import React from "react";
```

```
import ButtonClick from "./ButtonClick"; // Importing functional component
import ClickEventClass from "./ClickEventClass"; // Importing class component

const App = () => {
  return (
    <div className="App">
      <h1>Handling Click Events in React</h1>
      <ButtonClick />
      <ClickEventClass />
    </div>
  );
};

export default App;
```

✔ How It Works

1. **Using `onClick` in Functional and Class Components**
   - Functional: `onClick={handleClick}` directly calls a function.
   - Class: Requires `this.handleClick` and **binding in the constructor**.
2. **Passing Parameters in Event Handlers**
   - Arrow functions allow us to pass parameters:

```
jsx
CopyEdit
onClick={() => handleClickWithParam("Manuel")}
```

3. **Preventing Default Behavior**
   - Example: Preventing a link from redirecting:

```
jsx
CopyEdit
<a href="#" onClick={(e) => e.preventDefault()}>
  Click me (Prevents Redirect)
</a>
```

4. **Event Bubbling and Stopping Propagation**
   - Example: Stopping event propagation inside nested elements:

```
jsx
CopyEdit
<div onClick={() => console.log("Parent Clicked!")}>
  <button onClick={(e) => e.stopPropagation()}>
    Click Me (Stops Propagation)
  </button>
</div>
```

✔ Why Use Click Events in React?

Feature	Benefit
Interactive UIs	Enables user interactions like button clicks, form submissions, and navigation.
Dynamic Event Handling	Allows passing parameters and executing functions dynamically.
Improved User Experience	Makes the application responsive and engaging.

🚀 Real-World Use Cases for Click Events

- **Navigation**: Clicking on menu items to change views.
- **Modals and Dialogs**: Clicking a button to open/close a modal.
- **Forms**: Submitting a form when clicking a submit button.
- **Dynamic UI Updates**: Changing styles, animations, or content dynamically