Create a React Application “eventexamplesapp” to handle various events of the form elements in HTML.

1. Create “Increment” button to increase the value of the counter and “Decrement” button to decrease the value of the counter. The “Increase” button should invoke multiple methods.
   1. To increment the value
   2. Say Hello followed by a static message.

A screenshot of a computer

AI-generated content may be incorrect.

1. Create a button “Say Welcome” which invokes the function which takes “welcome” as an argument.

A screenshot of a computer

AI-generated content may be incorrect.

1. Create a button which invokes synthetic event “OnPress” which display “I was clicked”

A screenshot of a computer

AI-generated content may be incorrect.

Create a “CurrencyConvertor” component which will convert the Indian Rupees to Euro when the Convert button is clicked.

Handle the Click event of the button to invoke the handleSubmit event and handle the conversion of the euro to rupees.

A screenshot of a computer

AI-generated content may be incorrect.

**Solution**

Step 1 : create react project

npx create-react-app eventexamplesapp

cd eventexamplesapp

code .

Step 2 : App.js in src

// src/App.js

import React, { useState } from "react";

import CurrencyConverter from "./CurrencyConverter";

import "./App.css";

function App() {

  // counter state and message state

  const [count, setCount] = useState(0);

  const [message, setMessage] = useState("");

  // (a) increment function

  const increment = () => {

    setCount((prev) => prev + 1);

  };

  // (b) say hello followed by static message

  const sayHello = () => {

    setMessage("Hello — This is a static message!");

  };

  // Increase button should invoke multiple methods:

  // we call increment() and sayHello() from one wrapper

  const handleIncrease = () => {

    increment();

    sayHello();

  };

  // Decrement function

  const handleDecrement = () => {

    setCount((prev) => prev - 1);

  };

  // 2. Say Welcome -> function that takes an argument

  const sayMessage = (msg) => {

    setMessage(msg);

  };

  // 3. Synthetic "onPress" example:

  // Note: In web React we use onClick (React Native uses onPress).

  // We'll provide a handler named handlePress to simulate an onPress behavior.

  const handlePress = (e) => {

    // e is a React SyntheticEvent

    setMessage("I was clicked");

    console.log("Synthetic event type:", e.type); // shows "click"

  };

  return (

    <div className="app">

      <h1>Event Examples App</h1>

      <section className="card">

        <h2>Counter</h2>

        <p className="counter-value">Count: {count}</p>

        <div className="buttons-row">

          <button onClick={handleIncrease}>Increase</button>

          <button onClick={handleDecrement}>Decrement</button>

        </div>

        <small className="hint">

          Clicking <b>Increase</b> runs two methods: increment() and sayHello()

        </small>

      </section>

      <section className="card">

        <h2>Say Welcome</h2>

        {/\* Call function with "welcome" argument \*/}

        <button onClick={() => sayMessage("welcome")}>Say Welcome</button>

      </section>

      <section className="card">

        <h2>Simulated onPress (synthetic event)</h2>

        {/\* In web React, use onClick. We name handler handlePress \*/}

        <button onClick={handlePress}>Press Me</button>

      </section>

      <section className="card">

        <h2>Message (shows output of handlers)</h2>

        <p className="message">{message || "No message yet."}</p>

      </section>

      {/\* Currency Converter component (separate file) \*/}

      <CurrencyConverter />

    </div>

  );

}

export default App;

Step 3: create CurrencyConverter.js in src

// src/CurrencyConverter.js

import React, { useState } from "react";

/\*\*

 \* CurrencyConverter:

 \* - Converts INR <-> EUR

 \* - The form submission invokes handleSubmit (button click / form submit)

 \* - Uses a sample static rate (you can update the rate as needed)

 \*/

export default function CurrencyConverter() {

  const [amount, setAmount] = useState("");

  const [direction, setDirection] = useState("INRtoEUR"); // or "EURtoINR"

  const [result, setResult] = useState(null);

  // Sample conversion rate: 1 EUR = 90 INR (example)

  // Update this value if you want a different rate.

  const INR\_PER\_EUR = 90;

  // handleSubmit invoked on button click (or form submit)

  const handleSubmit = (e) => {

    e.preventDefault(); // stop page reload

    const value = parseFloat(amount);

    if (isNaN(value)) {

      setResult("Please enter a valid number.");

      return;

    }

    if (direction === "INRtoEUR") {

      const converted = value / INR\_PER\_EUR;

      setResult(`${value.toFixed(2)} INR = ${converted.toFixed(2)} EUR (rate: 1 EUR = ${INR\_PER\_EUR} INR)`);

    } else {

      const converted = value \* INR\_PER\_EUR;

      setResult(`${value.toFixed(2)} EUR = ${converted.toFixed(2)} INR (rate: 1 EUR = ${INR\_PER\_EUR} INR)`);

    }

  };

  return (

    <div className="card converter">

      <h2>Currency Converter</h2>

      <form onSubmit={handleSubmit}>

        <label>

          Amount:

          <input

            type="text"

            value={amount}

            onChange={(e) => setAmount(e.target.value)}

            placeholder="Enter amount"

          />

        </label>

        <div className="radio-row">

          <label>

            <input

              type="radio"

              name="direction"

              checked={direction === "INRtoEUR"}

              onChange={() => setDirection("INRtoEUR")}

            />

            INR → EUR

          </label>

          <label>

            <input

              type="radio"

              name="direction"

              checked={direction === "EURtoINR"}

              onChange={() => setDirection("EURtoINR")}

            />

            EUR → INR

          </label>

        </div>

        <button type="submit">Convert</button>

      </form>

      {result && <p className="result">{result}</p>}

      <p className="note">Using sample rate: 1 EUR = {INR\_PER\_EUR} INR. Update rate as needed.</p>

    </div>

  );

}

Step 4 : App.css in src

/\* src/App.css \*/

body {

  font-family: "Segoe UI", Tahoma, Geneva, Verdana, sans-serif;

  background: #f7f7f7;

  color: #222;

  padding: 20px;

}

.app {

  max-width: 800px;

  margin: 0 auto;

}

h1 {

  text-align: center;

  margin-bottom: 18px;

}

.card {

  background: white;

  padding: 16px;

  border-radius: 6px;

  box-shadow: 0 1px 4px rgba(0,0,0,0.08);

  margin-bottom: 16px;

}

.buttons-row > button,

.card button {

  margin-right: 8px;

  margin-top: 8px;

  padding: 8px 14px;

  font-size: 14px;

  cursor: pointer;

}

.counter-value {

  font-size: 20px;

  font-weight: 700;

}

.message {

  padding: 10px;

  background: #fafafa;

  border-radius: 4px;

}

.converter input[type="text"] {

  margin-left: 8px;

  padding: 6px;

  width: 160px;

}

.radio-row {

  margin: 8px 0;

}

.result {

  margin-top: 12px;

  font-weight: 700;

}

.note {

  margin-top: 8px;

  font-size: 12px;

  color: #555;

}

Step 5 : Run

npm start

**Output**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**