**CODE**

**public/index.html**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8" />

<link rel="icon" href="%PUBLIC\_URL%/favicon.ico" />

<meta name="viewport" content="width=device-width, initial-scale=1" />

<meta name="theme-color" content="#000000" />

<meta

name="description"

content="Web site created using create-react-app"

/>

<link rel="apple-touch-icon" href="%PUBLIC\_URL%/logo192.png" />

<!--

manifest.json provides metadata used when your web app is installed on a

user's mobile device or desktop. See https://developers.google.com/web/fundamentals/web-app-manifest/

-->

<link rel="manifest" href="%PUBLIC\_URL%/manifest.json" />

<!--

Notice the use of %PUBLIC\_URL% in the tags above.

It will be replaced with the URL of the `public` folder during the build.

Only files inside the `public` folder can be referenced from the HTML.

Unlike "/favicon.ico" or "favicon.ico", "%PUBLIC\_URL%/favicon.ico" will

work correctly both with client-side routing and a non-root public URL.

Learn how to configure a non-root public URL by running `npm run build`.

-->

<title>React App</title>

</head>

<body>

<noscript>You need to enable JavaScript to run this app.</noscript>

<div id="root"></div>

<!--

This HTML file is a template.

If you open it directly in the browser, you will see an empty page.

You can add webfonts, meta tags, or analytics to this file.

The build step will place the bundled scripts into the <body> tag.

To begin the development, run `npm start` or `yarn start`.

To create a production bundle, use `npm run build` or `yarn build`.

-->

</body>

</html>

**src/index.js**

import React from 'react';

import ReactDOM from 'react-dom/client';

import './index.css';

import App from './App';

import reportWebVitals from './reportWebVitals';

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(

<React.StrictMode>

<App />

</React.StrictMode>

);

// If you want to start measuring performance in your app, pass a function

// to log results (for example: reportWebVitals(console.log))

// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals

reportWebVitals();

**src/App.js**

import React from 'react';

import './App.css';

import EventExamples from './EventExamples';

import CurrencyConvertor from './CurrencyConvertor';

function App() {

return (

<div className="App">

<header className="App-header">

<h1>React Event Handling Demo</h1>

<EventExamples />

<CurrencyConvertor />

</header>

</div>

);

}

export default App;

**src/EventExamples.js**

import React, { Component } from 'react';

class EventExamples extends Component {

constructor(props) {

super(props);

this.state = {

counter: 0,

};

// Binding 'this' for the methods

this.incrementCounter = this.incrementCounter.bind(this);

this.decrementCounter = this.decrementCounter.bind(this);

this.sayHello = this.sayHello.bind(this);

this.handleMultipleIncrements = this.handleMultipleIncrements.bind(this);

}

// Method to increment the counter

incrementCounter() {

this.setState({ counter: this.state.counter + 1 });

}

// Method to decrement the counter

decrementCounter() {

this.setState({ counter: this.state.counter - 1 });

}

// Method to show a static message

sayHello() {

alert('Hello! The counter is being increased.');

}

// Method that invokes multiple other methods

handleMultipleIncrements() {

this.incrementCounter();

this.sayHello();

}

// Method that takes an argument

sayMessage(message) {

alert(message);

}

// Method to handle the synthetic event click

handleSyntheticEvent(event) {

// The 'event' object is a SyntheticEvent

console.log(event);

alert('I was clicked');

}

render() {

return (

<div style={{ border: '2px solid #61DAFB', padding: '20px', margin: '20px', borderRadius: '8px' }}>

<h2>Counter</h2>

<p>Current Count: {this.state.counter}</p>

{/\* Button to invoke multiple methods for incrementing \*/}

<button onClick={this.handleMultipleIncrements}>Increase</button> {/\* [cite: 19] \*/}

{/\* Button to decrease the counter value \*/}

<button onClick={this.decrementCounter}>Decrement</button>

<hr style={{ margin: '20px 0' }} />

{/\* Button that passes an argument to its handler \*/}

<button onClick={() => this.sayMessage('welcome')}>Say Welcome</button> {/\* [cite: 22] \*/}

<hr style={{ margin: '20px 0' }} />

{/\* Button to demonstrate a synthetic event \*/}

<button onClick={this.handleSyntheticEvent}>OnPress</button> {/\* [cite: 23] \*/}

</div>

);

}

}

export default EventExamples;

**src/CurrencyConverter.js**

import React, { Component } from 'react';

class CurrencyConvertor extends Component {

constructor(props) {

super(props);

this.state = {

inr: '',

euro: null,

};

this.handleInputChange = this.handleInputChange.bind(this);

this.handleSubmit = this.handleSubmit.bind(this);

}

handleInputChange(event) {

this.setState({ inr: event.target.value });

}

handleSubmit(event) {

event.preventDefault();

const inrValue = parseFloat(this.state.inr);

if (!isNaN(inrValue)) {

// Assuming an exchange rate, e.g., 1 EUR = 90 INR

const euroValue = inrValue / 90;

this.setState({ euro: euroValue.toFixed(2) });

} else {

this.setState({ euro: 'Invalid input' });

}

}

render() {

return (

<div style={{ border: '2px solid #61DAFB', padding: '20px', margin: '20px', borderRadius: '8px' }}>

<h2>Currency Convertor</h2>

<form onSubmit={this.handleSubmit}>

<div>

<label>

Indian Rupees (INR):

<input

type="text"

value={this.state.inr}

onChange={this.handleInputChange}

style={{ marginLeft: '10px' }}

/>

</label>

</div>

<button type="submit" style={{ marginTop: '10px' }}>Convert</button>

</form>

{this.state.euro !== null && (

<p>

Equivalent in Euro (EUR): €{this.state.euro}

</p>

)}

</div>

);

}

}

export default CurrencyConvertor;

**src/App.css**

.App {

text-align: center;

}

.App-header {

background-color: #282c34;

min-height: 100vh;

display: flex;

flex-direction: column;

align-items: center;

justify-content: center;

font-size: calc(10px + 2vmin);

color: white;

}

button {

background-color: #61DAFB;

border: none;

color: #282c34;

padding: 10px 20px;

text-align: center;

text-decoration: none;

display: inline-block;

font-size: 16px;

margin: 4px 2px;

cursor: pointer;

border-radius: 4px;

font-weight: bold;

}

button:hover {

background-color: #21a1f1;

}

input {

padding: 8px;

border-radius: 4px;

border: 1px solid #ccc;

}

**OUTPUT**

