Experiment 08

# **Aim:** Create the front-end application using JSX, Components, Props, State, Forms, Events, Routers, Refs, Keys of React.

# **Requirements:** Chrome, VsCode

## **Theory:**

# **1) JSX**

JSX stands for JavaScript XML. It allows us to write HTML in React and makes it easier to write and add HTML in React. JSX allows us to write HTML elements in JavaScript and place them in the DOM without any createElement()  and/or appendChild() methods.

JSX converts HTML tags into react elements.

ReactDOM.render(myelement, document.getElementById('root'));

# **2) Components**

Components are independent and reusable bits of code. They serve the same purpose as JavaScript functions, but work in isolation and return HTML. Components come in two types, Class components and Function components, in this tutorial we will concentrate on Function components.

### Class Component

A class component must include the extends React.Component statement. This statement creates an inheritance to React.Component, and gives your component access to React.Component's functions.

The component also requires a render() method, this method returns HTML.

### Example

Create a Class component called Car

class Car extends React.Component {

render() {

return <h2>Hi, I am a Car!</h2>;

}

}

### Function Component

Here is the same example as above, but created using a Function component instead.

A Function component also returns HTML, and behaves much the same way as a Class component, but Function components can be written using much less code, are easier to understand, and will be preferred in this tutorial.

### Example

Create a Function component called Car

function Car() {

return <h2>Hi, I am a Car!</h2>;

}

# **3) Props**

React Props are like function arguments in JavaScript *and* attributes in HTML.

To send props into a component, use the same syntax as HTML attributes:

### Example

Add a "brand" attribute to the Car element:

const myelement = <Car brand="Ford" />;

The component receives the argument as a props object:

### Example

Use the brand attribute in the component:

function Car(props) {

return <h2>I am a { props.brand }!</h2>;

}

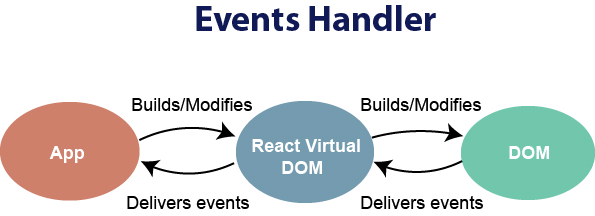
# **4) State**

The state is an instance of React Component Class can be defined as an object of a set of observable properties that control the behavior of the component. In other words, the State of a component is an object that holds some information that may change over the lifetime of the component. For example, let us think of the clock that we created in [this article](https://www.geeksforgeeks.org/reactjs-rendering-elements/), we were calling the render() method every second explicitly, but React provides a better way to achieve the same result and that is by using State, storing the value of time as a member of the component’s state. We will look into this more elaborately later in the article.

# **6) Events**

An event is an action that could be triggered as a result of the user action or system generated event. For example, a mouse click, loading of a web page, pressing a key, window resizes, and other interactions are called events.

React has its own event handling system which is very similar to handling events on DOM elements. The react event handling system is known as Synthetic Events. The synthetic event is a cross-browser wrapper of the browser's native event.



# **7) Routers**

React Router is a standard library for routing in React. It enables the navigation among views of various components in a React Application, allows changing the browser URL, and keeps the UI in sync with the URL.

# **8) Refs**

Refs is the shorthand used for references in React. It is similar to keys in React. It is an attribute which makes it possible to store a reference to particular DOM nodes or React elements. It provides a way to access React DOM nodes or React elements and how to interact with it. It is used when we want to change the value of a child component, without making the use of props.

In React, Refs can be created by using React.createRef(). It can be assigned to React elements via the ref attribute. It is commonly assigned to an instance property when a component is created, and then can be referenced throughout the component.

1. class MyComponent extends React.Component {
2. constructor(props) {
3. super(props);
4. this.callRef = React.createRef();
5. }
6. render() {
7. return <div ref={this.callRef} />;
8. }
9. }

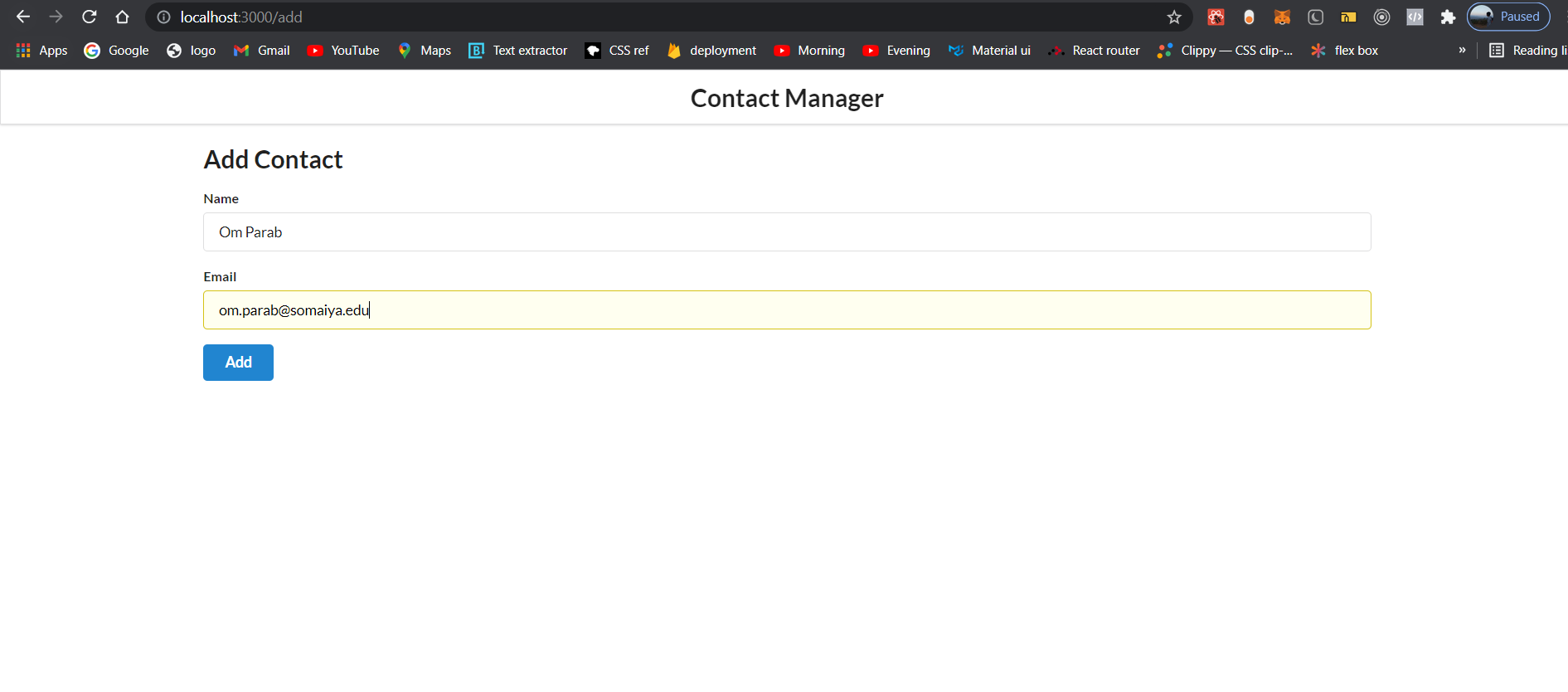
# **9) Keys**

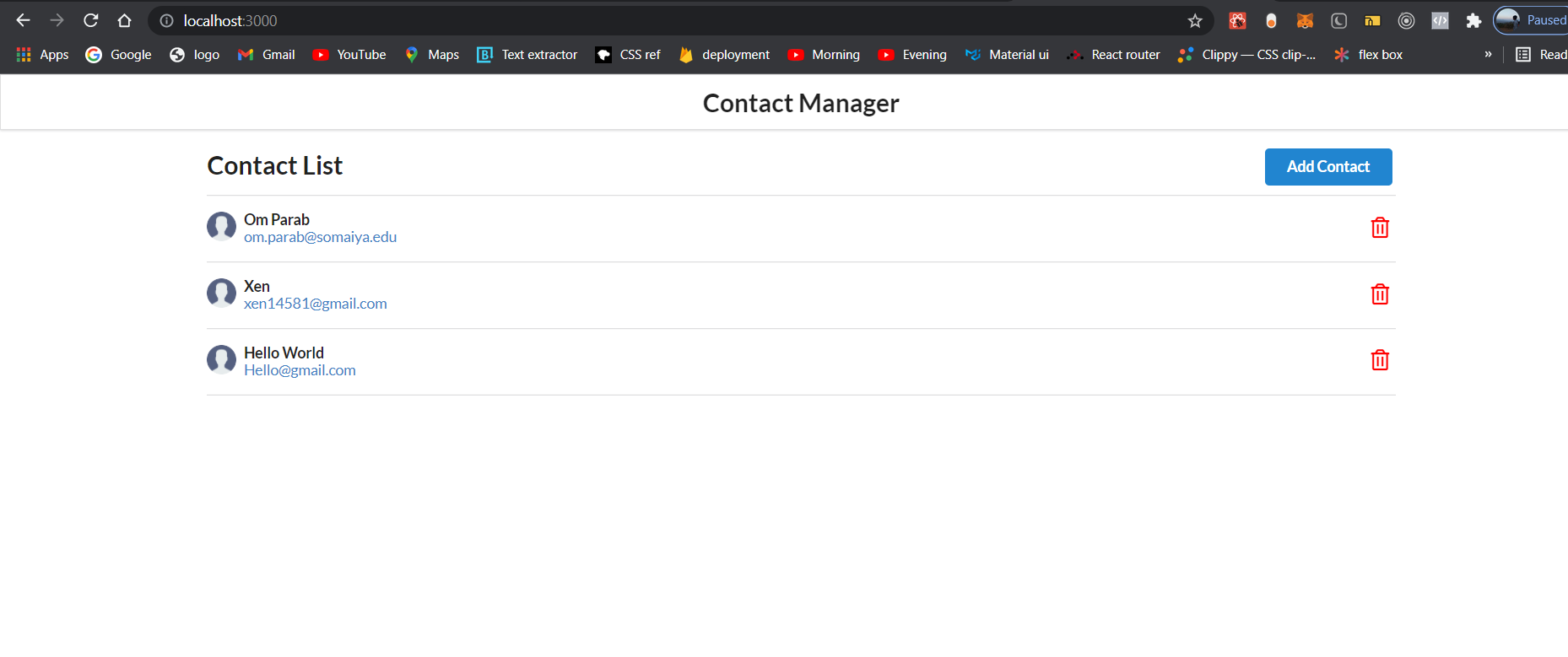
React keys are useful when working with dynamically created components or when your lists are altered by the users. Setting the key value will keep your components uniquely identified after the change.

## Using Keys

Let's dynamically create Content elements with unique index (i). The map function will create three elements from our data array. Since the key value needs to be unique for every element, we will assign i as a key for each created element.

# **Output:**





# **Conclusion:** ReactJS provides flexibility and easiness to make websites interactive responsive and fast. It is one of the best Javascript library for front end development .

**References**: <https://www.w3schools.com>

**CODE-LINK:** https://drive.google.com/drive/folders/1HLdhQQmKGw5CWGAmXL0ECFWYViNXVu74?usp=sharing