**Define Props**

In React, *props* (short for "properties") are read-only inputs passed from a parent component to a child component. They allow components to be dynamic and reusable by accepting values such as strings, numbers, objects, or even functions. Props are passed using attributes in JSX and help maintain a unidirectional data flow in the application. Inside the receiving component, props can be accessed using props.propertyName or directly via destructuring. Since props are immutable within the child component, they ensure consistency and avoid unintended changes to data coming from higher levels in the component tree.

**Explain Default Props**

*Default props* are predefined values assigned to props when no value is explicitly passed by the parent component. They ensure that the component still functions properly and displays meaningful data even if some props are missing. In older versions of React, default props for class components were defined using the defaultProps property. For function components, default values can be set using default parameter syntax or by assigning ComponentName.defaultProps.

**Example using function component:**

function Greeting({ name = "Guest" }) {

return <h1>Hello, {name}!</h1>;

}

If Greeting is used without a name prop, it will default to "Guest".

**Identify the differences between State and Props**

Props and state are both plain JavaScript objects used in React to control component behavior, but they serve different purposes. Props are passed from parent to child and are immutable within the child component. They are used to configure components and help in rendering dynamic content based on external input. State, on the other hand, is managed internally within a component and can change over time. State is mutable and is typically used for interactive or time-sensitive data such as form inputs, button clicks, or API responses. In short, props are for external data input, while state is for internal data management.

**Explain reactDOM.render()**

The ReactDOM.render() method is used to render a React component or element into the actual DOM. It connects your React application to a specific part of the HTML page, usually a <div> with an id like 'root'. In React versions before 18, this method was used to mount the root component.

**Syntax:**

ReactDOM.render(<App />, document.getElementById('root'));

This code tells React to render the App component inside the DOM element with the id 'root'. It replaces the contents of that element with the virtual DOM structure returned by App. In React 18+, this has been replaced with ReactDOM.createRoot().render() for improved concurrent rendering capabilities, but the core concept remains: it bridges React components with the actual browser DOM.

**Index.js :-**

import React from 'react';

import ReactDOM from 'react-dom/client';

import OnlineShopping from './OnlineShopping';

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

root.render(

  <React.StrictMode>

    <OnlineShopping />

  </React.StrictMode>

);

Cart.js:-

class Cart {

  constructor(itemname, price) {

    this.itemname = itemname;

    this.price = price;

  }

}

export default Cart;

**OnlineShopping.js :-**

import React from 'react';

import Cart from './Cart';

class OnlineShopping extends React.Component {

  constructor(props) {

    super(props);

    this.items = [

      new Cart('Laptop', 80000),

      new Cart('TV', 120000),

      new Cart('Washing Machine', 50000),

      new Cart('Mobile', 30000),

      new Cart('Fridge', 70000)

    ];

  }

  render() {

    return (

      <div style={{ textAlign: 'center', marginTop: '50px' }}>

        <h1 style={{ color: 'green' }}>Items Ordered :</h1>

        <div style={{ display: 'inline-block', border: '1px solid grey', padding: '10px' }}>

          <table style={{ borderCollapse: 'collapse', color: 'seagreen' }}>

            <thead>

              <tr>

                <th style={{ border: '1px solid grey', padding: '8px' }}>Name</th>

                <th style={{ border: '1px solid grey', padding: '8px' }}>Price</th>

              </tr>

            </thead>

            <tbody>

              {this.items.map((item, index) => (

                <tr key={index}>

                  <td style={{ border: '1px solid grey', padding: '8px' }}>{item.itemname}</td>

                  <td style={{ border: '1px solid grey', padding: '8px' }}>{item.price}</td>

                </tr>

              ))}

            </tbody>

          </table>

        </div>

      </div>

    );

  }

}

export default OnlineShopping;

Output :-



