**1. WHAT IS ReactJS**

ReactJS is an **open-source JavaScript library** developed by Facebook (now Meta) for building dynamic and responsive user interfaces, especially single-page applications (SPAs).

**Key Features:**

* **Component-Based:** UI is built using independent reusable pieces called components.
* **Declarative:** Focus on “what to render” rather than “how”.
* **Virtual DOM:** React updates only the changed parts of the UI for better performance.
* **One-Way Data Binding:** Data flows from parent to child components.
* **JSX:** A syntax extension that allows writing HTML-like code inside JavaScript.

**2. SETTING UP REACT**

Before working with React, ensure you have **Node.js** and **npm** installed.

**Create a new React app**

npx create-react-app myapp

cd myapp

npm start

This will start a local development server and open your app at http://localhost:3000/.

**3. UNDERSTANDING JSX (JavaScript XML)**

JSX allows you to write HTML elements inside JavaScript.  
React then converts this JSX into JavaScript using a compiler like Babel.

**Example (JSX vs JS):**

Without JSX:

const element = React.createElement('h1', null, 'Hello React');

With JSX:

const element = <h1>Hello React</h1>;

**JSX Rules:**

* You can only return **one root element**.
* Use **className** instead of class.
* Expressions can be embedded using {}.

Example:

const name = "Udaya";

const element = <h2>Hello, {name}! Welcome to React.</h2>;

**4. Components in React**

A **component** is a reusable block of code that controls a part of the UI.

**Types of Components**

1. **Functional Components** – simple functions returning JSX.
2. **Class Components** – ES6 classes extending React.Component.

**Functional Component Example**

function Welcome() {

return <h2>Welcome to Dhruv Compusoft Training</h2>;

}

export default Welcome;

**Rendering the Component**

In App.js:

import Welcome from './Welcome';

function App() {

return (

<div>

<Welcome />

</div>

);

}

**Output:**  
Welcome to Dhruv Compusoft Training

**Class Component Example**

import React, { Component } from 'react';

class WelcomeClass extends Component {

render() {

return <h2>Hello from Class Component</h2>;

}

}

export default WelcomeClass;

**5. Props (Passing Data Between Components)**

**Props** (short for *properties*) are used to pass data from parent to child components.

**Parent Component:**

function App() {

return <Student name="Udaya" course=".NET" />;

}

**Child Component (Student.js):**

function Student(props) {

return (

<div>

<h3>Student Name: {props.name}</h3>

<p>Course: {props.course}</p>

</div>

);

}

export default Student;

**Output:**

Student Name: Udaya

Course: .NET

**Props are immutable**, meaning you cannot modify them inside the child component.

**6. State (Component Data Management)**

**State** is used to store dynamic, changeable data in a component.  
When state changes, the component re-renders automatically.

**Using useState Hook (in Functional Components)**

import { useState } from 'react';

function Counter() {

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

return (

<div>

<h3>Count: {count}</h3>

<button onClick={() => setCount(count + 1)}>Increase</button>

<button onClick={() => setCount(count - 1)}>Decrease</button>

</div>

);

}

export default Counter;

**Explanation:**

* useState(0) initializes state variable count with value 0.
* setCount is a function to update count.
* Clicking buttons updates state and re-renders UI.

**7. COMBINING MULTIPLE COMPONENTS**

function App() {

return (

<div>

<Welcome />

<Student name="Udaya" course=".NET Full Stack" />

<Counter />

</div>

);

}

Each component handles a small part of the app, making code modular and easier to maintain.

**8. COMPONENT HIERARCHY**

Example:

App.js

├── Header.js

├── Student.js

└── Counter.js

This structure ensures a **clean, scalable** React application.

**9. BEST PRACTICES**

* Always capitalize component names (e.g., Student, not student).
* Keep each component in its own file.
* Use props for data passing and useState for internal state.
* Keep components small and focused on one task.
* Use meaningful variable and function names.

**Mini Practice Task (Day 4)**

**Objective:** Create a simple React app with three components:

1. Header.js – Displays page title.
2. Student.js – Displays student info using props.
3. Counter.js – Demonstrates use of useState.

**App Structure:**

/src

├── App.js

├── Header.js

├── Student.js

└── Counter.js

**Task:**

* Display “Dhruv Compusoft Training Portal” as heading.
* Pass props from App to Student (name, course).
* Create a counter with increment and decrement buttons.

A screenshot of a computer program

AI-generated content may be incorrect.**Snapshots :**

Code: App.js

A screenshot of a computer program

AI-generated content may be incorrect.

Code: Counter.js

A screenshot of a computer

AI-generated content may be incorrect.

Code: Header.js

A screenshot of a computer

AI-generated content may be incorrect.

Code: index.js

A screenshot of a computer

AI-generated content may be incorrect.

Code: Student.js

A screenshot of a computer

AI-generated content may be incorrect.

Terminal Commands

A screenshot of a computer

AI-generated content may be incorrect.

Output: Counter Increment

A screenshot of a computer

AI-generated content may be incorrect.

Output: Counter Decrement