**Composing components**

Composing components is a fundamental concept in ReactJS that refers to the practice of building complex user interfaces by combining and nesting smaller, reusable components.

It allows you to break down your UI into manageable, self-contained pieces, making your codebase more organized, maintainable, and easier to develop and scale.

**1. Components in React:**

In React, a component is a self-contained, reusable building block for creating user interfaces.

Components can be thought of as functions that return elements (typically JSX) describing what should be displayed on the screen.

Components can represent both UI elements (e.g., buttons, forms) and more complex structures (e.g., a chat window or an entire application).

**2. The Need for Composing Components:**

Building modern web applications often involves creating complex user interfaces with numerous features and sections.

Composing components allows developers to manage this complexity by breaking the UI into smaller, manageable pieces.

Each component can focus on a specific aspect of the UI or a particular functionality, making the codebase more maintainable and understandable.

**3. Benefits of Composing Components:**

Reusability: You can reuse components throughout your application, reducing duplication of code and effort.

Modularity: Components are modular and can be developed, tested, and maintained independently.

Separation of Concerns: Each component is responsible for a specific concern, promoting the separation of concerns principle.

Easy Maintenance: Smaller, focused components are easier to debug and maintain than monolithic code.

Scalability: As your application grows, you can easily add new components and integrate them into your UI.

**4. Component Composition:**

In React, components can be composed by nesting one component inside another within the JSX code.

This nesting allows you to create a hierarchy of components where parent components can contain child components, and those child components can contain other child components, forming a tree-like structure.

Props (short for properties) are used to pass data from parent components to child components, allowing them to communicate and share information.

**Simple Examples**

**Step 1: Create a React App**

Create a React application using Create React App or your preferred method. Navigate to your project directory and install the necessary dependencies:

**npx create-react-app Sample**

**cd Sample**

**Step 2: Create Necessary Components**

**Steps 3: Compose and Render Components**

**Step 4: Running the Application**

**npm start**

See your React application at [http://localhost:3000](http://localhost:3000/), displaying the output of the developed application.

**1.A simple "Header" component within a "Main" component.**

// Header.js

import React from 'react';

function Header() {

return <h1>Welcome to My App</h1>;

}

export default Header;

// Main.js

import React from 'react';

import Header from './Header';

function Main() {

return (

<div>

<Header />

<p>This is the main content of my app.</p>

</div>

);

}

export default Main;

// App.js

import React from 'react';

import Main from './Main';

function App() {

return (

<div>

<Main />

</div>

);

}

export default App;

**2. Passing Data between Parent and Child Components:**

// Parent.js

import React from 'react';

import Child from './Child';

function Parent() {

const message = "Hello from Parent Component";

return (

<div>

<Child message={message} />

</div>

);

}

export default Parent;

// Child.js

import React from 'react';

function Child(props) {

return <p>{props.message}</p>;

}

export default Child;

// App.js

import React from 'react';

import Parent from './Parent';

function App() {

return (

<div>

<Parent />

</div>

);

}

export default App;

**3. Conditional Rendering with Components:**

import React from 'react';

function App() {

const isLoggedIn = true;

return (

<div>

{isLoggedIn ? <WelcomeUser /> : <LoginPrompt />}

</div>

);

}

function WelcomeUser() {

return <h1>Welcome, User!</h1>;

}

function LoginPrompt() {

return <p>Please log in to continue.</p>;

}

export default App;

**4. Mapping Data to Components:**

import React from 'react';

function App() {

const users = [

{ id: 1, name: 'John' },

{ id: 2, name: 'Jane' },

{ id: 3, name: 'Bob' },

];

return (

<div>

<h1>User List</h1>

{users.map((user) => (

<User key={user.id} name={user.name} />

))}

</div>

);

}

function User(props) {

return <p>{props.name}</p>;

}

export default App;

**Application Name: Employee Directory**

**Program Statement:**

You are tasked with building an Employee Directory web application using ReactJS. The goal of this application is to create a user-friendly interface for managing and displaying employee information. To achieve this, you'll need to compose different components to create a cohesive user interface.

**Application Requirements:**

1. Employee List Component (EmployeeList): This component should render a list of employees, displaying their names and job titles. The list should be scrollable if there are many employees.
2. Employee Detail Component (EmployeeDetail): When a user clicks on an employee's name in the list, this component should display detailed information about the selected employee, including their name, photo, email, and phone number.
3. Search Component (SearchBar): Implement a search bar at the top of the employee list to allow users to search for employees by name. The list should dynamically update as the user types in the search bar.
4. Sorting Component (SortButton): Add buttons that allow users to sort the employee list by name or job title in ascending or descending order.
5. Add Employee Form Component (AddEmployeeForm): Create a form that allows users to add new employees to the directory. Collect information such as name, job title, email, and phone number.

**//App.js**

import React, { useState } from 'react';

import './App.css';

import EmployeeList from './EmployeeList';

import EmployeeDetail from './EmployeeDetail';

import SearchBar from './SearchBar';

function App() {

const initialEmployees = [

{

id: 1,

name: 'John Doe',

jobTitle: 'Software Developer',

email: 'john@example.com',

phone: '123-456-7890',

},

{

id: 2,

name: 'Jane Smith',

jobTitle: 'Designer',

email: 'jane@example.com',

phone: '987-654-3210',

},

];

const [employees, setEmployees] = useState(initialEmployees);

const [selectedEmployee, setSelectedEmployee] = useState(null);

const [searchTerm, setSearchTerm] = useState('');

// Function to handle employee selection

const handleEmployeeSelect = (employee) => {

setSelectedEmployee(employee);

};

// Function to handle search

const handleSearch = (searchTerm) => {

setSearchTerm(searchTerm);

};

// Filter employees based on search term

const filteredEmployees = employees.filter((employee) =>

employee.name.toLowerCase().includes(searchTerm.toLowerCase())

);

return (

<div className="App">

<h1>Employee Directory</h1>

<SearchBar onSearch={handleSearch} />

<div className="App-content">

<EmployeeList

employees={filteredEmployees}

onSelect={handleEmployeeSelect}

/>

<EmployeeDetail employee={selectedEmployee} />

</div>

</div>

);

}

export default App;

**//EmployeeList.js:**

import React from 'react';

function EmployeeList({ employees, onSelect }) {

return (

<div className="EmployeeList">

<h2>Employees</h2>

<ul>

{employees.map((employee) => (

<li key={employee.id} onClick={() => onSelect(employee)}>

{employee.name} - {employee.jobTitle}

</li>

))}

</ul>

</div>

);

}

export default EmployeeList;

**//EmployeeDetail.js:**

import React from 'react';

function EmployeeDetail({ employee }) {

return (

<div className="EmployeeDetail">

<h2>Employee Details</h2>

{employee ? (

<div>

<p>Name: {employee.name}</p>

<p>Job Title: {employee.jobTitle}</p>

<p>Email: {employee.email}</p>

<p>Phone: {employee.phone}</p>

</div>

) : (

<p>Select an employee to view details</p>

)}

</div>

);

}

export default EmployeeDetail;

**SearchBar.js:**

import React, { useState } from 'react';

function SearchBar({ onSearch }) {

const [searchTerm, setSearchTerm] = useState('');

const handleSearchChange = (e) => {

const value = e.target.value;

setSearchTerm(value);

onSearch(value);

};

return (

<div className="SearchBar">

<input

type="text"

placeholder="Search employees by name"

value={searchTerm}

onChange={handleSearchChange}

/>

</div>

);

}

export default SearchBar;