**Explain the need and Benefits of React Context API**

**Need and Benefits of React Context API**

**What is the Problem?**

In React, data is usually passed from parent to child using **props**. But when multiple nested components need access to the same data (like user info, theme, or language), **prop drilling** becomes a problem:

Passing data through many intermediate components—even if they don't need it—just to reach the ones that do.

**What is React Context API?**

The **React Context API** is a built-in solution to manage and share global data (like app theme, authentication status, etc.) across components without prop drilling.

**Benefits:**

| **Benefit** | **Description** |
| --- | --- |
| Centralized Data | Store app-wide values like user auth, theme, language, etc. |
| Avoids Prop Drilling | No need to pass props through every level of the component tree. |
| Cleaner Code | Easier to read and maintain. |
| Works with Functional Components | Compatible with React Hooks like useContext. |
| Dynamic Updates | Consumers re-render automatically when context values change. |

**Working with createContext()**

// 1. Create context

import React, { createContext, useContext, useState } from 'react';

const ThemeContext = createContext(); // createContext()

// 2. Provider component

function ThemeProvider({ children }) {

const [theme, setTheme] = useState('light');

return (

<ThemeContext.Provider value={{ theme, setTheme }}>

{children}

</ThemeContext.Provider>

);

}

// 3. Use context in any child component

function ThemedComponent() {

const { theme, setTheme } = useContext(ThemeContext);

return (

<div>

<p>Current theme: {theme}</p>

<button onClick={() => setTheme('dark')}>Switch to Dark</button>

</div>

);

}

// 4. Use it in App

function App() {

return (

<ThemeProvider>

<ThemedComponent />

</ThemeProvider>

);

}

**List the types of Router Components**

| **Component** | **Purpose** |
| --- | --- |
| BrowserRouter | Uses HTML5 history API (pushState) for clean URLs (e.g., /about) |
| HashRouter | Uses URL hash (#) for routing (e.g., /#/about) — better for older browsers |
| MemoryRouter | Stores history in memory (not the address bar) — ideal for testing or non-browser environments |
| StaticRouter | Used for server-side rendering (SSR), especially in frameworks like Next.js |
| NativeRouter | Used in React Native, part of react-router-native |

**Developers of Apps Centric Solutions have created an employee management application which supports light and dark themes for the buttons. The current solution uses the react state and props to provide the theme name to be used from App component to Employee List component and from there to Employee Card component. Quality assurance team analyzed the solutions and found the technique being used to be a substandard one. React architect suggested to use the react context API to share the theme name with nested child components instead of passing them down using props from the parent component.**

**Index.js :-**

import React from 'react';

import ReactDOM from 'react-dom';

import './index.css';

import App from './App';

import reportWebVitals from './reportWebVitals';

ReactDOM.render(

  <React.StrictMode>

    <App />

  </React.StrictMode>,

  document.getElementById('root')

);

reportWebVitals();

**App.js :-**

// src/App.js

import React, { useState } from 'react';

import EmployeesList from './EmployeesList';

import ThemeContext from './ThemeContext';

function App() {

  const [theme, setTheme] = useState('light');

  return (

    <ThemeContext.Provider value={theme}>

      <div className="App">

        <h1>Employee Management</h1>

        {/\* Toggle Theme (optional for testing) \*/}

        <button onClick={() => setTheme(theme === 'light' ? 'dark' : 'light')}>

          Toggle Theme

        </button>

        <EmployeesList />

      </div>

    </ThemeContext.Provider>

  );

}

export default App;

**Employee.js :-**

class Employee {

    constructor(id, name, email, phone){

        this.id=id;

        this.name=name;

        this.email=email;

        this.phone=phone;

    }

}

const EmployeesData=[

    new Employee(101,'Jojo','jojo@congizant.com','98238971234'),

    new Employee(102,'Sam','sam@congizant.com', '9981184126'),

    new Employee(103,'Elisa','elisa@cognizant.com','9989389735')

];

export default Employee;

export {EmployeesData};

**EmployeeCard.js :-**

// src/EmployeeCard.js

import React, { useContext } from 'react';

import ThemeContext from './ThemeContext';

function EmployeeCard({ employee }) {

  const theme = useContext(ThemeContext);

  return (

    <div className={`employee-card ${theme}`}>

      <h3>{employee.name}</h3>

      <p>{employee.position}</p>

      <button className={theme}>View Profile</button>

    </div>

  );

}

export default EmployeeCard;

**EmployeesList.js :-**

// src/EmployeesList.js

import React from 'react';

import EmployeeCard from './EmployeeCard';

const employees = [

  { id: 1, name: 'John Doe', position: 'Developer' },

  { id: 2, name: 'Jane Smith', position: 'Designer' },

];

function EmployeesList() {

  return (

    <div>

      <h2>Employee List</h2>

      {employees.map(emp => (

        <EmployeeCard key={emp.id} employee={emp} />

      ))}

    </div>

  );

}

export default EmployeesList;

**Output :-**



