**Question 1: What is the Context API in React? How is it used to manage global state across multiple components?**

Answer:

**What is the Context API?**  
The Context API is a built-in feature in React that provides a way to share data (state) across multiple components without having to pass props down manually through every level of the component tree. It's designed to solve the problem of "prop drilling" - passing props through intermediate components that don't need them, just to get them to deeply nested components that do.

**How it's used to manage global state:**

Create a Context: You create a context object that will hold the shared data.

Provide the Context: You wrap your component tree with a Context Provider that supplies the shared data.

Consume the Context: Any component within the Provider tree can access the shared data without needing props.

**When to use Context:**

Theme data (light/dark mode)

User authentication status

Preferred language (i18n)

Any data that many components need access to

When NOT to use Context:

For data that only a few components need (use props instead)

For frequently updated data (use state management libraries like Redux instead)

As a replacement for all component state

**Question 2: Explain how createContext() and useContext() are used in React for sharing state.**

Answer:

Step-by-Step Implementation:

1. Create Context with createContext()

// contexts/ThemeContext.js

import { createContext } from 'react';

// Create a context with a default value

const ThemeContext = createContext({

theme: 'light',

toggleTheme: () => {}, // Default function

});

export default ThemeContext;

2. Provide Context with a Provider Component

// App.js

import { useState } from 'react';

import ThemeContext from './contexts/ThemeContext';

import Header from './components/Header';

import MainContent from './components/MainContent';

function App() {

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

// Function to toggle between light and dark theme

const toggleTheme = () => {

setTheme(prevTheme => prevTheme === 'light' ? 'dark' : 'light');

};

// Context value that will be shared

const contextValue = {

theme: theme,

toggleTheme: toggleTheme

};

return (

// Provide the context to all children components

<ThemeContext.Provider value={contextValue}>

<div className={`app ${theme}`}>

<Header />

<MainContent />

</div>

</ThemeContext.Provider>

);

}

export default App;

3. Consume Context with useContext() Hook

// components/Header.js

import { useContext } from 'react';

import ThemeContext from '../contexts/ThemeContext';

function Header() {

// Access the context value

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

return (

<header style={{

backgroundColor: theme === 'light' ? '#f0f0f0' : '#333',

color: theme === 'light' ? '#000' : '#fff',

padding: '1rem'

}}>

<h1>My Application</h1>

<button onClick={toggleTheme}>

Switch to {theme === 'light' ? 'Dark' : 'Light'} Mode

</button>

</header>

);

}

export default Header;

Another component consuming the same context:

// components/MainContent.js

import { useContext } from 'react';

import ThemeContext from '../contexts/ThemeContext';

function MainContent() {

const { theme } = useContext(ThemeContext);

return (

<main style={{

backgroundColor: theme === 'light' ? '#fff' : '#222',

color: theme === 'light' ? '#000' : '#fff',

padding: '2rem',

minHeight: '400px'

}}>

<h2>Welcome to our app!</h2>

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

<p>This content automatically updates when the theme changes.</p>

</main>

);

}

export default MainContent;

Alternative: Using Context.Consumer (Class Components)

jsx

// For class components (without hooks)

function Header() {

return (

<ThemeContext.Consumer>

{({ theme, toggleTheme }) => (

<header style={{

backgroundColor: theme === 'light' ? '#f0f0f0' : '#333',

color: theme === 'light' ? '#000' : '#fff'

}}>

<h1>My Application</h1>

<button onClick={toggleTheme}>

Switch to {theme === 'light' ? 'Dark' : 'Light'} Mode

</button>

</header>

)}

</ThemeContext.Consumer>

);

}