Lesson 05 Demo 01

Creating a Theme Button Using Context API

Objective: To develop a React App which uses the concept of Context API

Tools Required: Node Terminal, React app, and Visual Studio Code

Prerequisites: Knowledge of creating a React app and understanding of the folder structure

Steps to be followed:

1. Create a new React app
2. Implement the Toolbar.js function
3. Wrap the Toolbar component in a ThemeProvider component
4. Update App.js to use the ThemeProvider component
5. Run the app

Step 1: Create a new React app

1.1 Open the terminal and run the **npx create-react-app context-demo** command

```
shreemayeebhatt@ip-172-31-22-250:~$ npx create-react-app context-demo
```

Note: This command will create a new **React** app with the name **context-demo**

1.2 Run the **cd context-demo** command in the terminal

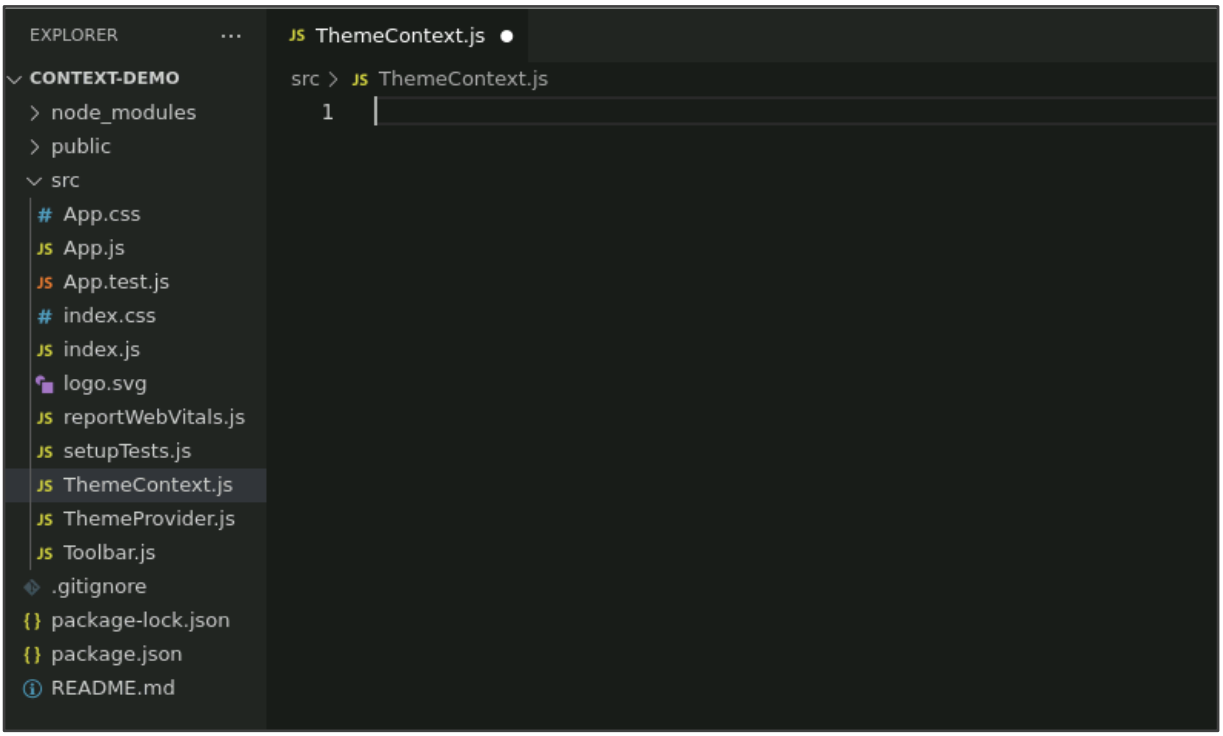
```
Happy hacking!
shreemayeebhatt@ip-172-31-22-250:~$ cd context-demo/
```

Note: This will change the current directory to the newly created **React** app directory

1.3 Open **Visual Studio Code** in Simplilearn lab

1.4 Open the folder to navigate to the **context-demo** directory

1.5 Inside the **src** directory, create the **ThemeContext.js** file



1.6 Import React in the **ThemeContext.js** file

1.7 Use the **React.createContext()** method to create a new context called **ThemeContext** and provide a default value of **light** in parameter

1.8 Export the **ThemeContext** object to default

```
import React from 'react';

const ThemeContext = React.createContext('light');

export default ThemeContext;
```

```
//ThemeContext.js

import React, { createContext } from 'react';
const ThemeContext = createContext();
export default ThemeContext;
```

Step 2: Implement the Toolbar.js function

2.1 Create the **Toolbar.js** file in the **src** directory

2.2 Import **React** and the **ThemeContext** from the **ThemeContext.js** file

```
import React from 'react';
import ThemeContext from './ThemeContext';
```

2.3 Create a functional component named called **Toolbar**

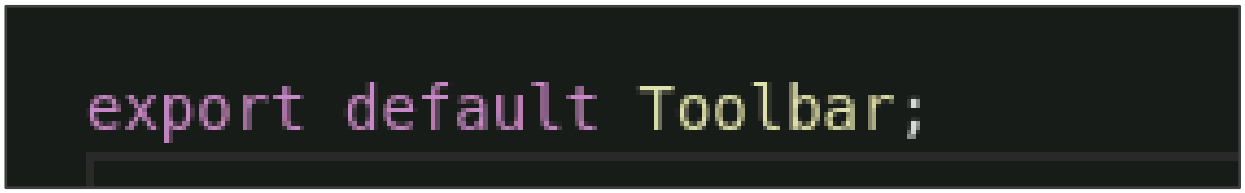
```
const Toolbar = () => {
```

2.4 In the **Toolbar** component, return the **JSX** elements

```
  return (
    <div>
      <h2>Toolbar</h2>
      <button
        style={{ backgroundColor: theme }}
        onClick={toggleTheme}
      >
        Change Theme
      </button>
    </div>
  );
};
```

2.5 Render the desired content inside the **ThemeContext.Consumer** component, which will have access to the current theme value

2.6 Export the **Toolbar** component




```
//Toolbar.js
import React, { useContext } from 'react';
import ThemeContext from './ThemeContext';
const Toolbar = () => {
const themeContext = useContext(ThemeContext);
const { theme, toggleTheme } = themeContext;
return (
<div>
<h2>Toolbar</h2>
 <button
style={{ backgroundColor: theme }}
onClick={toggleTheme}
 >
Change Theme
</button>
 </div>
 );
};

export default Toolbar;
```

Step 3: Wrap the Toolbar component in a ThemeProvider component

3.1 Open the **App.js** file from the **src** directory

3.2 Import **React** and **Toolbar** component, and the **ThemeContext** from the **ThemeContext.js** file

```
import React from 'react';
import Toolbar from './Toolbar';
import ThemeContext from './ThemeContext';
```

3.3 Create the **App** functional component

3.4 Inside the **App** component, return **JSX** elements that include the **ThemeContext.Provider** component

3.5 Set the value property of the **ThemeContext.Provider** component to the desired theme value

3.6 Render the **Toolbar** component inside the **ThemeContext.Provider** component

3.7 Export the **App** component

```
const App = () => {
  return (
    <div>
      <h1>Context Demo</h1>
      <ThemeProvider>
        <Toolbar />
      </ThemeProvider>
    </div>
  );
};

export default App;
```

```
import React from 'react';
import Toolbar from './Toolbar';
import ThemeProvider from './ThemeContext';

const App = () => {
return (
<div>
<h1>Context Demo</h1>
<ThemeProvider>
 <Toolbar />
 </ThemeProvider>
</div>
);
};
export default App;
```

Note: This will be a new component that will set the theme for our app

3.8 In the **src** directory, create a new file named **ThemeProvider.js**

3.9 Import **React** and **ThemeContext** from the **ThemeContext.js** file

```
import React from 'react';
import ThemeContext from './ThemeContext';
```

3.10 Create the **ThemeProvider** class component that extends **React.Component**

3.11 In the **ThemeProvider** component, define a state property named **theme** with an initial value of **light**

```
const ThemeProvider = ({ children }) => {
  const [theme, setTheme] = useState('light');
```

3.12 Implement the **toggleTheme** method that updates the state to toggle between light and dark

```
const toggleTheme = () => {
  setTheme(prevTheme => (prevTheme === 'light' ? 'dark' : 'light'));
};
```

3.13 In the render method, return **JSX** elements that include the **ThemeContext.Provider** Component

3.14 Set the value property of the **ThemeContext.Provider** component to the current value of the **theme** state

3.15 Render the desired content inside the **ThemeContext.Provider** component, including a button that triggers the **toggleTheme** method

3.16 Export the **ThemeProvider** component

//ThemeProvider.js

```
import React, { useState, useEffect } from 'react';
import ThemeContext from './ThemeContext';

const ThemeProvider = ({ children }) => {
const [theme, setTheme] = useState('light');
```

```
useEffect(() => {
const body = document.querySelector('body');

body.style.backgroundColor = theme === 'light' ? '#f0f0f0' : '#333333';
}, [theme]);
const toggleTheme = () => {
setTheme(prevTheme => (prevTheme === 'light' ? 'dark' : 'light'));
};

const themeContextValue = {
theme,
 toggleTheme,
 };
return (
<ThemeContext.Provider value={themeContextValue}>
{children}
</ThemeContext.Provider>
);
};
export default ThemeProvider;
```

S

Step 4: Update App.js to use the ThemeProvider component

4.1 In the **App.js** file, import the **ThemeProvider** component

4.2 Wrap the **Toolbar** component inside the **ThemeProvider** component

```
import React from 'react';
import Toolbar from './Toolbar';
import ThemeProvider from './ThemeProvider';

const App = () => {
return (
<div>
<h1>Context Demo</h1>
<ThemeProvider>
<Toolbar />
</ThemeProvider>
</div>
```

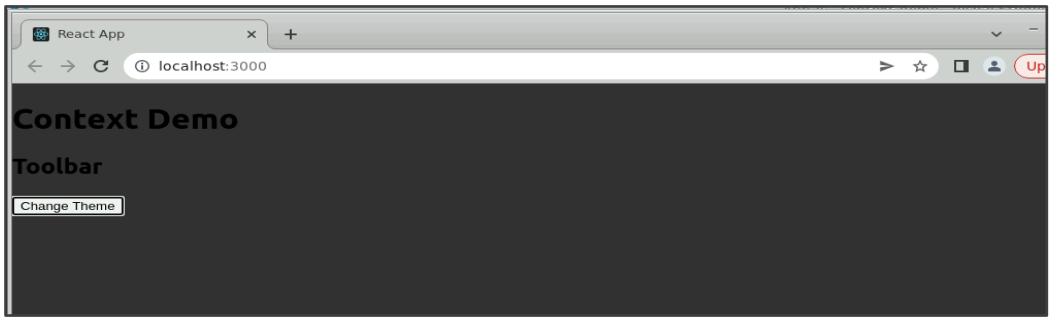
```
  );
};
export default App;
```

Step 5: Run the app

5.1 In the terminal, navigate to the project directory

5.2 Run the **npm start** command to start the app

5.3 Open your browser and navigate to **http://localhost:3000**



In conclusion, this demo showcased how to create a theme button using the Context API in React, allowing for a centralized theme management system in your application. The Context API provides a convenient way to share state and functionality across multiple components without the need for prop drilling.

With this, you have successfully implemented a theme button using the Context API in React, providing a centralized theme management system for your application.