

Lesson 04 Demo 04

Creating a React Application Using the useContext Hook

Objective: To develop a simple app with a button that toggles the theme between light and dark

Tools required: Node terminal, React app, and Visual Studio Code

Prerequisites: Knowledge of a React app and the folder structure

Steps to be followed:

1. Create a new React app
2. Create the ThemeContext.js file
3. Implement the ChildComponent.js file
4. Modify the App.js file
5. Run the app and check the functionality

Step 1: Create a new React app

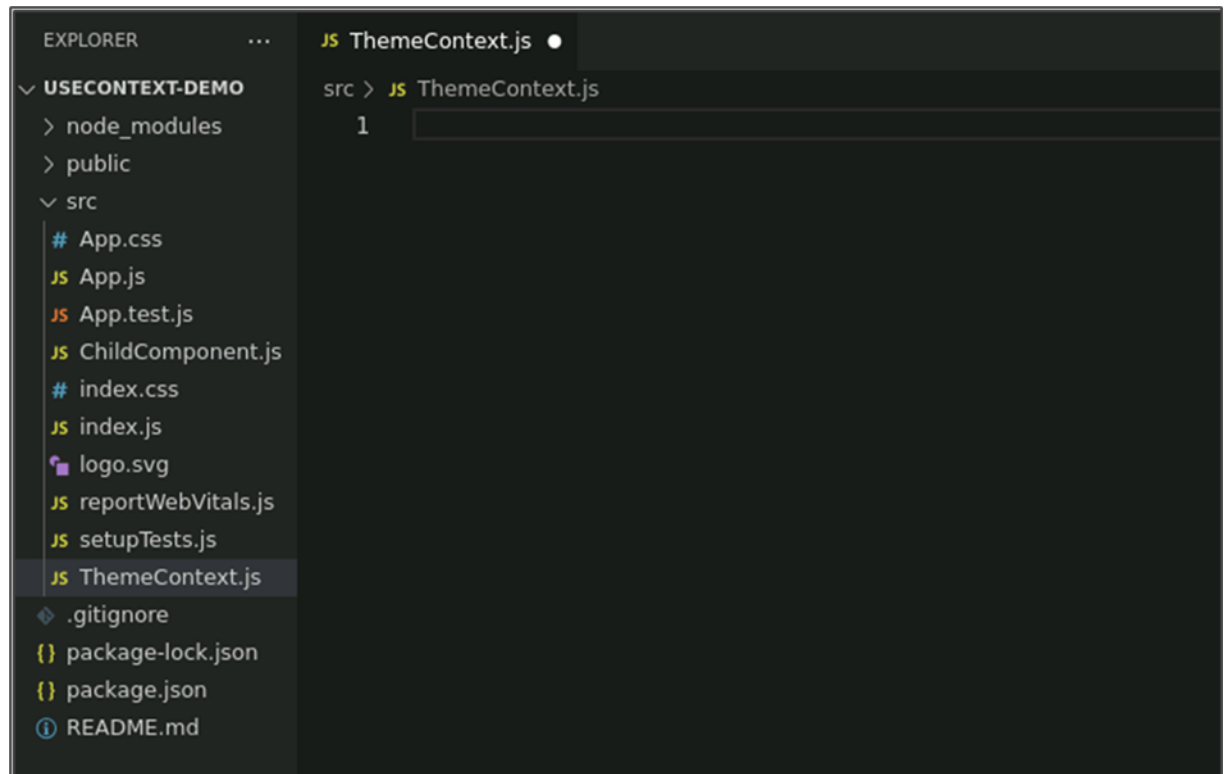
- 1.1 Open your terminal and run the **npm create-react-app usecontext-demo** command to create a new React app with the name **usecontext-demo**

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

- 1.2 Move to the newly created project directory by running the **cd usecontext-demo** command

Step 2: Create the ThemeContext.js file

2.1 Open the React app **usecontext-demo** with **Visual Studio Code** editor to create the **ThemeContext.js** file within the **src** directory



2.2 Inside **ThemeContext.js**, file import the **createContext** function from React using the following code:

```
// ThemeContext.js
import { createContext } from 'react';
const ThemeContext = createContext();
export default ThemeContext;
```

```
src > js ThemeContext.js > ...
1   import { createContext } from 'react';
2
3   const ThemeContext = createContext();
4
5   export default ThemeContext;
```

Step 3: Implement the ChildComponent.js file

3.1 In your code editor, create a new file called **ChildComponent.js** in the **src** directory

3.2 Import the **useContext** Hook from **React** and the **ThemeContext** file using the following code:

```
/ ChildComponent.js
import React, { useContext } from 'react';
import ThemeContext from './ThemeContext';
```

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

3.3 Inside the **ChildComponent** function component, use the **useContext** Hook to access the theme value from the **ThemeContext** using the following code:

```
function ChildComponent() {
  const theme = useContext(ThemeContext);

  return <p>Current Theme: {theme}</p>;
}
```

```
function ChildComponent() {  
  const theme = useContext(ThemeContext);  
  return <p>Current Theme: {theme}</p>;  
}
```

Note: Refer to the following code to configure the **ChildComponent.js** file:

```
//ChildComponent.js  
import React, { useContext } from 'react';  
import ThemeContext from './ThemeContext';  
  
function ChildComponent() {  
  const theme = useContext(ThemeContext);  
  return <p>Current Theme: {theme}</p>;  
}  
export default ChildComponent;
```

```
src > JS ChildComponent.js > ...  
1  import React, { useContext } from 'react';  
2  import ThemeContext from './ThemeContext';  
3  
4  function ChildComponent() {  
5    const theme = useContext(ThemeContext);  
6    return <p>Current Theme: {theme}</p>;  
7  }  
8  
9  export default ChildComponent;
```

Step 4: Modify the App.js file

- 4.1 Open the **App.js** file in your code editor located in the **src** directory and add the code given below:

```
// App.js
import React, { useState, useContext } from 'react';
import ThemeContext from './ThemeContext';
import ChildComponent from './ChildComponent';
```

```
src > JS App.js > App
1  import React, { useState } from 'react';
2  import ThemeContext from './ThemeContext';
3  import ChildComponent from './ChildComponent';
4
```

- 4.2 Inside the **App** component, declare a state variable **theme** using the **useState** Hook and set its initial value to **light**

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

```
function App() {
  const [theme, setTheme] = useState('light');
```

- 4.3 Define a function **toggleTheme** that will toggle the theme between **light** and **dark** when the function is called

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

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

4.4 Update the `document.body.style.backgroundColor` based on the theme as shown in the screenshot:

```
document.body.style.backgroundColor = theme === 'dark' ? 'black' : 'white';
```

4.5 Enter the following code to return the **JSX markup** for the **App** component

```
return (  
  <div>  
    <ThemeContext.Provider value={theme}>  
      <h1>App</h1>  
      <button onClick={toggleTheme}>Toggle Theme</button>  
      <ChildComponent />  
    </ThemeContext.Provider>  
  </div>  
  
  );  
}
```

```
return (  
  <div>  
    <ThemeContext.Provider value={theme}>  
      <h1>App</h1>  
      <button onClick={toggleTheme}>Toggle Theme</button>  
      <ChildComponent />  
    </ThemeContext.Provider>  
  </div>  
  
  );  
}
```

Note: Refer to the following code snippet to configure the App.js file:

```
//App.js
import React, { useState } from 'react';
import ThemeContext from './ThemeContext';
import ChildComponent from './ChildComponent';

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

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

  // Update the document body background color based on the theme
  document.body.style.backgroundColor = theme === 'dark' ? 'black' : 'white';

  return (
    <div>
      <ThemeContext.Provider value={theme}>
        <h1>App</h1>
        <button onClick={toggleTheme}>Toggle Theme</button>

        <ChildComponent />
      </ThemeContext.Provider>
    </div>
  );
}

export default App;
```

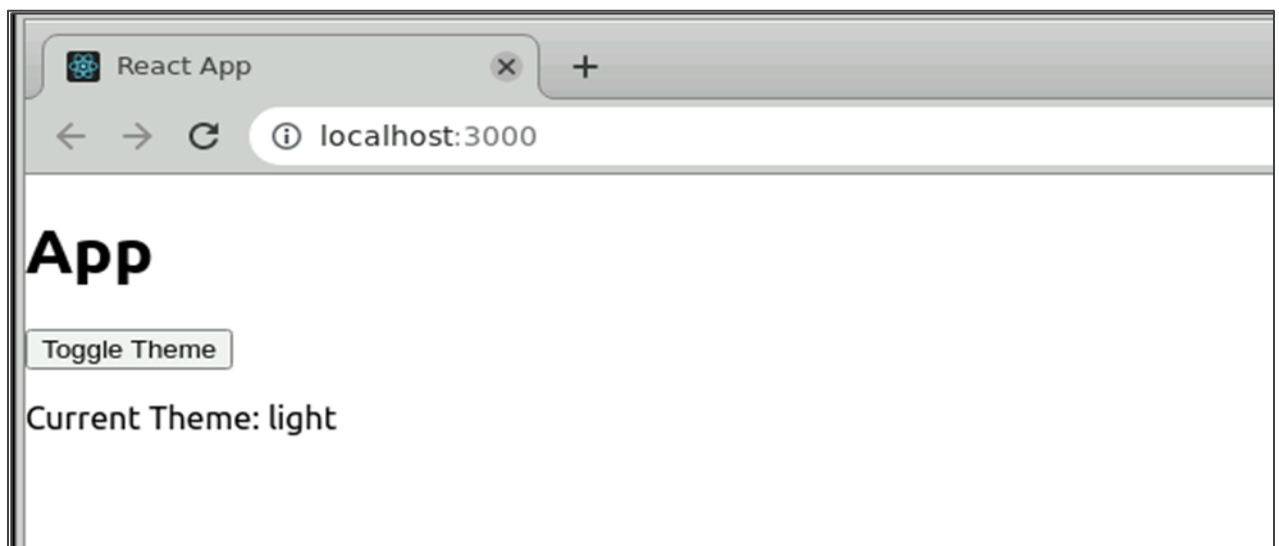
Step 5: Run the app and check the functionality

5.1 In your terminal, execute the following command to start the development server and run the app:

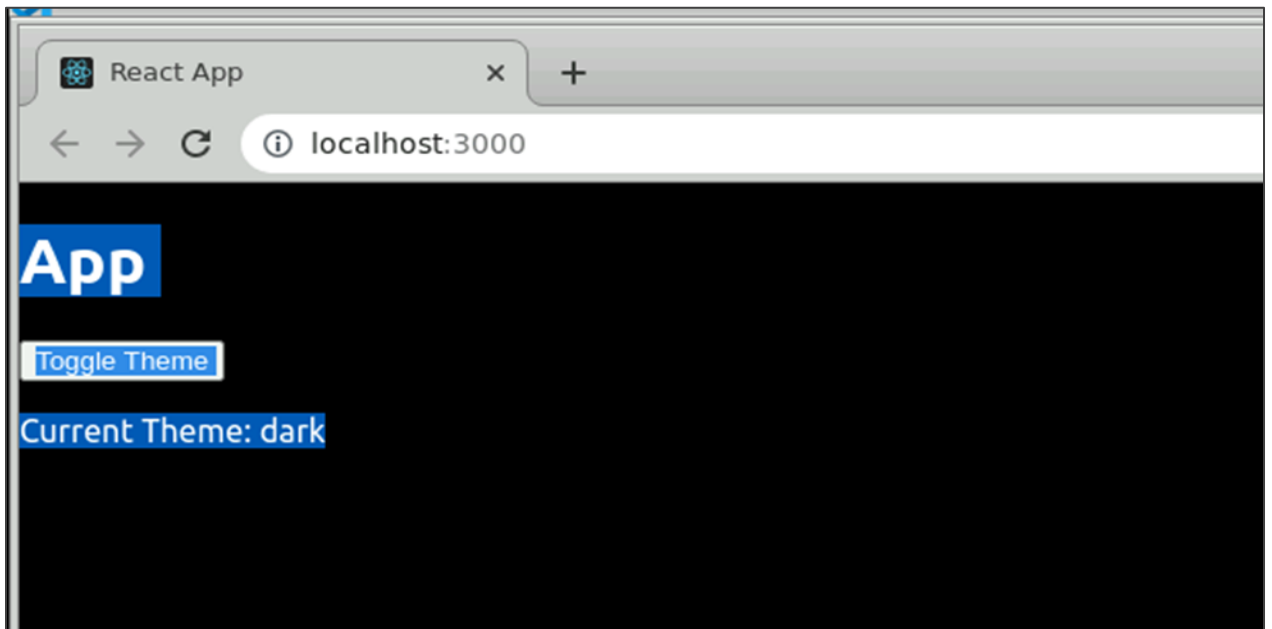
npm start

5.2 Once the server starts successfully, open **http://localhost:3000** in your browser to view the app

Light Theme:



Dark Theme:



In the browser, you should see a simple app with the heading **App**, a button labeled as **Toggle Theme** and the **ChildComponent**. Click the **Toggle Theme** button to switch the theme between light and dark. The **ChildComponent** will display the current theme using the **useContext** Hook.

With this, you have successfully created a simple app with a button that toggles the theme between light and dark and demonstrates the working of the **useContext** Hook.