

## Task 8: React Context for State Management in a Profile Website

### Objective:

Learn to use the React Context API for managing state in a profile website. This task includes creating and providing a context to manage the theme (light/dark mode) and consuming the context in components.

### Pre-requisites:

- Basic understanding of HTML, CSS, and JavaScript
- Familiarity with a code editor like Visual Studio Code
- Basic knowledge of React

### Concepts Covered:

- Understanding Context API
- Creating and Providing a Context
- Consuming Context in Components

### Concepts:

#### 1. Understanding Context API:

The Context API in React is a powerful tool for managing global state. It allows you to share data across all levels of the application without passing props down manually at every level.

#### 2. Creating and Providing a Context:

Create a `ThemeContext` to manage and toggle the theme of your profile website (light/dark mode).

```
// src/context/ThemeContext.js
import React, { createContext, useState } from 'react';

const ThemeContext = createContext();

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

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

  return (
    <ThemeContext.Provider value={{ theme, toggleTheme }}>
      {children}
    </ThemeContext.Provider>
  );
};

export { ThemeContext, ThemeProvider };
```

### 3. Consuming Context in Components:

Use the `useContext` hook in the `Header` component to toggle and display the current theme.

```
// src/components/Header.js
import React, { useContext } from 'react';
import { ThemeContext } from '../context/ThemeContext';

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

  return (
    <header style={{ background: theme === 'light' ? '#fff' : '#333', color:
theme === 'light' ? '#000' : '#fff' }}>
      <h1>Profile Website</h1>
      <button onClick={toggleTheme}>
        Switch to {theme === 'light' ? 'Dark' : 'Light'} Mode
      </button>
    </header>
  );
};

export default Header;
```

#### Setup:

##### 1. Install Visual Studio Code (VS Code):

Download and install VS Code from [Visual Studio Code](#).

##### 2. React Developer Tools:

Install the React Developer Tools browser extension for [Chrome](#) or [Firefox](#).

##### 3. Git:

Install Git for version control. You can download it from [Git](#).

##### 4. Node.js and npm:

Ensure Node.js and npm are installed on your machine. You can download and install them from [Node.js](#).

##### 5. Setting Up the React Project:

- Open your terminal.
- Navigate to the directory where you want to create your project.
- Run the following command to create a new React project:

```
npx create-react-app profile-website
```

##### 6. Creating Context Directory:

- Navigate to the `src` directory and create a new directory named `context`.

- Inside the `context` directory, create a file named `ThemeContext.js`.

## Tasks:

### Part 1: Create and Provide a Context (30 minutes)

#### 1. Create a `ThemeContext` to manage and toggle the theme of your profile website (light/dark mode):

- Example:

```
// src/context/ThemeContext.js
import React, { createContext, useState } from 'react';

const ThemeContext = createContext();

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

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

  return (
    <ThemeContext.Provider value={{ theme, toggleTheme }}>
      {children}
    </ThemeContext.Provider>
  );
};

export { ThemeContext, ThemeProvider };
```

- Goal: Learn to create a context and use the Provider to make it available across components.

### Part 2: Consume Context in Components (30 minutes)

#### 1. Use the `useContext` hook in the Header component to toggle and display the current theme:

- Example:

```
// src/components/Header.js
import React, { useContext } from 'react';
import { ThemeContext } from '../context/ThemeContext';

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

  return (
    <header style={{ background: theme === 'light' ? '#fff' : '#333',
      color: theme === 'light' ? '#000' : '#fff' }}>

```

```

        <h1>Profile Website</h1>
        <button onClick={toggleTheme}>
            Switch to {theme === 'light' ? 'Dark' : 'Light'} Mode
        </button>
    </header>
  );
};

export default Header;

```

## 2. Wrap your application in the ThemeProvider:

- Example:

```

// src/App.js
import React from 'react';
import { ThemeProvider } from '../context/ThemeContext';
import Header from '../components/Header';

const App = () => {
  return (
    <ThemeProvider>
      <Header />
      { /* Other components */ }
    </ThemeProvider>
  );
};

export default App;

```









### Instructions:

1. Write the required code in the appropriate files in the `src/context` and `src/components` directories.
2. Open the terminal and navigate to the project directory.
3. Run the React project using:

```
npm start
```

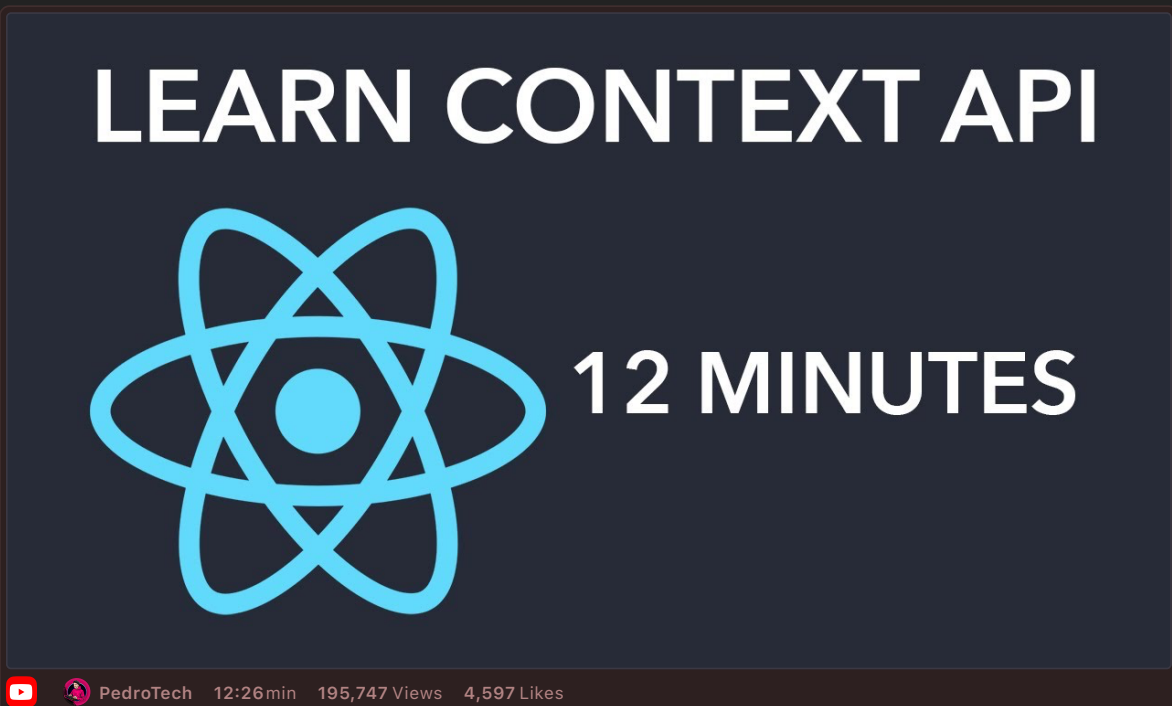
4. Open the project in a web browser to ensure the code displays correctly.
5. Use the browser's developer tools and React Developer Tools to debug and inspect the elements.

### Resources:

-  **Context – React**  
A JavaScript library for building user interfaces  
 <https://reactjs.org/docs/context.html>
-  **Getting Started – React**  
A JavaScript library for building user interfaces  
 <https://reactjs.org/docs/getting-started.html>
-  **Create React App**  
Set up a modern web app by running one command.  
 <https://create-react-app.dev/>
-  **Documentation for Visual Studio Code**  
Find out how to set-up and get the most from Visual Studio Code. Optimized for building and d...  
 <https://code.visualstudio.com/docs>

#### Videos:

-    
**Learn**  
**useContext**  
**In 13 Minutes**  
  Web Dev Simplified 13:08min 703,255 Views 14,688 Likes



#### GitHub Instructions:

1. **Open in Visual Studio Code:**

After clicking on the "Open in Visual Studio Code" button from the GitHub Classroom confirmation page, VSCode will open the repository directly. If prompted, select "Open" or "Allow" to open the repository in VSCode.

2. **Open the Terminal in VSCode:**

In VSCode, open a terminal by selecting Terminal > New Terminal from the top menu.

**3. Complete the Task:**

In VSCode, write your solution in the appropriate files in the `src/context` and `src/components` directories.

**4. Run and Test Your Code:**

Open your terminal, navigate to your project directory, and run:

```
npm start
```

**5. Commit Your Changes:**

In the VSCode terminal, add your changes to git:

```
git add src/context/* src/components/*
```

Commit your changes with a meaningful message:

```
git commit -m "Completed task 29"
```

**6. Push Your Changes to Your Repository:**

Push your changes to your forked repository:

```
git push origin main
```

**7. Create a Pull Request:**

Go to your repository on GitHub.

Click on the "Pull Requests" tab.

Click the "New Pull Request" button.

Ensure the base repository is the original template repository and the base branch is `main`.

Ensure the head repository is your forked repository and the compare branch is `main`.

Click "Create Pull Request".

Add a title and description for your pull request and submit it.

**Summary of Commands:**

```
# Open in Visual Studio Code

# Open terminal in VSCode

# Complete the task by editing src/context/* src/components/*

# Navigate to the project directory
cd profile-website

# Run your code
npm start

# Add, commit, and push your changes
git add src/context/* src/components/*
git commit -m "Completed task 8"
git push origin main

# Create a pull request on GitHub
```