Task 10: Advanced Styling in React for a Profile Website

Objective:

Learn to apply advanced styling techniques in a React application using CSS-in-JS libraries like Styled-Components or Emotion, and CSS Modules. This task focuses on converting existing components to use these advanced styling methods in the context of a profile website.

Pre-requisites:

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

Concepts Covered:

- Using Styled Components
- Implementing CSS Modules

Concepts:

1. **Using Styled Components:**

   Convert the styling of the Profile component to use Styled Components.

   ```
   // src/components/Profile.js
   import styled from 'styled-components';

   const ProfileContainer = styled.div`
       display: flex;
       flex-direction: column;
       align-items: center;
       background-color: ${props => (props.darkMode ? '#333' : '#fff')};
       color: ${props => (props.darkMode ? '#fff' : '#000')};
   `;

   const Profile = ({ darkMode, name, occupation }) => {
       return (
           <ProfileContainer darkMode={darkMode}>
               <h1>{name}</h1>
               <p>{occupation}</p>
           </ProfileContainer>
       );
   };
   ```

   Hints:

   - Use `styled-components` to create a styled container.
   - Apply dynamic styling based on props (e.g., darkMode).

2. **Implementing CSS Modules:**

   Refactor the ProductList component to use CSS Modules for styling.

```
/* src/components/ProductList.module.css */
.productList {
    display: flex;
    flex-wrap: wrap;
    gap: 20px;
}

.productItem {
    background-color: #f9f9f9;
    padding: 20px;
    border: 1px solid #ddd;
    border-radius: 5px;
}
```

```
// src/components/ProductList.js
import styles from './ProductList.module.css';

const ProductList = ({ products }) => {
    return (
        <div className={styles.productList}>
            {products.map(product => (
                <div key={product.id} className={styles.productItem}>
                    <h2>{product.name}</h2>
                    <p>{product.description}</p>
                </div>
            ))}
        </div>
    );
};
```

Hints:

- Use CSS Modules to create scoped styles.
- Import the CSS module and apply styles using the imported class names.

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. **Installing Additional Dependencies:**
    - Navigate to the project directory:

    ```
    cd profile-website
    ```

    - Install Styled Components:

    ```
    npm install styled-components
    ```

Tasks:

Part 1: Using Styled Components (30 minutes)

1. **Convert the styling of the Profile component to use Styled Components:**
    - Example hint:

    ```
    const ProfileContainer = styled.div`
        display: flex;
        flex-direction: column;
        align-items: center;
        background-color: ${props => (props.darkMode ? '#333' : '#fff')};
        color: ${props => (props.darkMode ? '#fff' : '#000')};
    `;
    ```

    - Goal: Understand how to apply dynamic and reusable styles using CSS-in-JS.

Part 2: Implementing CSS Modules (30 minutes)

1. **Refactor the ProductList component to use CSS Modules for styling:**
    - Example hint:

    ```
    .productList {
        display: flex;
        flex-wrap: wrap;
        gap: 20px;
    }

    .productItem {
        background-color: #f9f9f9;
        padding: 20px;
    ```

```
    border: 1px solid #ddd;
    border-radius: 5px;
}
```

```
import styles from './ProductList.module.css';
```

- Goal: Learn how CSS Modules provide scoped and modular styling in a React application.

Instructions:

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

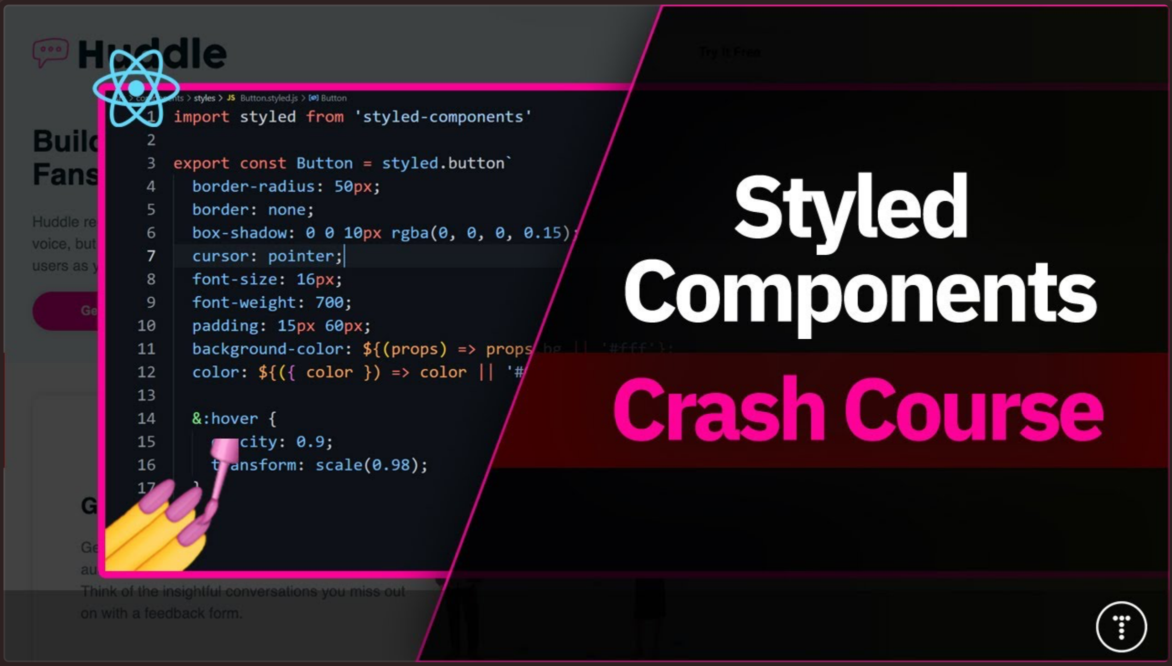
```
npm start
```

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:

- styled-components: Documentation
  Learn how to use styled-components and to style your apps without stress
  Author **styled-components**
- Github
  https://github.com/css-modules/css-modules
- 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:

- 

```
import styled from 'styled-components'

export const Button = styled.button`
  border-radius: 50px;
  border: none;
  box-shadow: 0 0 10px rgba(0, 0, 0, 0.15);
  cursor: pointer;
  font-size: 16px;
  font-weight: 700;
  padding: 15px 60px;
  background-color: ${(props) => props.bg || '#fff'};
  color: ${({ color }) => color || '#

  &:hover {
    city: 0.9;
    transform: scale(0.98);
```

  Traversy Media 48:23min 252,892 Views 6,545 Likes

-

  Adrian Twarog 15:30min 327,321 Views 7,041 Likes

-

Telmo Sampaio 10:25min 97,432 Views 2,076 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/components` directory.

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/components/*
   ```

   Commit your changes with a meaningful message:

```
git commit -m "Completed task 31"
```

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/components/*

# Navigate to the project directory
cd profile-website

# Run your code
npm start

# Add, commit, and push your changes
git add src/components/*
git commit -m "Completed task 10"
git push origin main

# Create a pull request on GitHub
```