Rachel Ruddy
Professor Tse
CS 396: Full Stack Development
19 Feb 2023

Mini-Project 1 Resubmission: Explanation and Defense

Overview

For the resubmission of Mini-Project 1, I have modified my three-page NextJS site to include the use of a Theme context as well as to clean up unnecessary and extraneous pieces of code. The site has a consistent header and footer across all three pages, and the theme toggle persists across all three pages during a given user session. In approaching my resubmission, I hoped to understand context through implementation in a way that I had been unable to achieve on the first submission.

Problem Statement

My goal is to enhance my personal portfolio site with cleaner, more concise code as well as a light/dark mode theme toggle maintains theme information across all three pages of the site via context and state handling.

Design Approach

Upon reapproaching this project, my first action was to address all stylistic/syntactical comments. These were the issues that were the quickest to fix, like changing my component tags to self-closing tags within the Layout file and removing commented-out code. Beyond these simple fixes, my primary objectives were to separate the input boxes of the Contact Form into their own component, and to connect the theme-toggle to the theme context to enable light-dark mode.

For the former, I created a basic component file called UserInput and isolated one instance of the repeated contact form input boxes. From this instance, I identified which aspects were shared among all instances (the broad structure) and which were unique to each instance (the id and the label associated with the input). These I passed into the UserInput component as props from the Contact component. Ultimately, I chose to keep the text *area* (where the body of the email message would go) separate from the instances of UserInput, for simplicity's sake.

The latter task took most of my focus during my revising period because it was a problem left unsolved upon my first submission of the project. I began to approach the issue by prioritizing generating a functional connection between the ThemeContext and the theme checkbox in the Header component. To reach this connection, I first imported the ThemeContextProvider into the layouts file and used it to wrap around the Header, children, and footer, thus allowing the Theme Context to be used within any location of the site. This required moderate debugging, as my import path was slightly incorrect.

Once the ThemeContextProvider was successfully imported and implemented, I continued by implementing useContext into my Header.js file to try to connect the theme checkbox to the toggleTheme method provided by the Theme Context. Facing similar issues as with the layout context provider import, I eventually was able to successfully import the theme and toggleTheme fields from the Theme Context, converting the Header component to a client-side component in doing so. From this point, my theme toggle was successfully able to change the visual theme of the header component.

I wanted to implement the context in a way that could modify the theme from the layout file, eliminating the need for repetition. Unfortunately, however, I was limited by time constraints and chose instead to convert my components and page files to be client side instead, granting them the ability to make use of the useClient function. From here, with some small modifications to the css, I was able to complete my light-dark mode implementation in such a way that persisted across all three pages of the app during a given user session.

With the new changes in this resubmission, my site has gone from using primarily SSR to being almost entirely CSR, a change I made to accommodate time constraints while still delivering worthwhile functionality. Though I faced similar challenges as before, from obscure bugs to a general lack of understanding, my final portfolio website demonstrates the understanding I have gained of the values and techniques of front-end development.