Personal Portfolio Website

Tyler Dempski

Overview/Background

Maintaining a personal portfolio website is a useful way of displaying accomplishments and past work without the constraints of a one page resume. Additionally, modern frameworks such as React.js/Next.js allow for development time for interesting websites to be drastically reduced compared to vanilla HTML/CSS/JS websites. Using the Next.js framework it is possible to develop an interesting, interactive display of my past work in a brief period of time

Problem Statement

The goal of this project is to design an interactive website using Next.js that effectively displays my past work while also providing several opportunities for user input, which must persist across navigation throughout the page.

Design Approach

The first stage of design was done with google drawings for simple wireframes. This allowed me to structure my project in a way that is reflective of the final design at the start, and decide ahead of time which components will be used.

For simplicity and readability, the top level Page.tsx is kept solely as navigation logic and component calls, acting as a view controller. All individual pages, including the navigation bar, are their own components. The work pages are particularly advantageous to be used as their own components, as they are structurally identical, so by switching on the navigation state it is possible to simply feed the template work page a different set of data to display.

The navigation bar handles the navigation state, taking the navigation state and change function as props, as there is only one level of abstraction, so it is unnecessary to use dependency injection. The buttons provide visual feedback when pressed, and there is always a hide/show navbar button visible on the left as a chevron pointing in the corresponding direction.

About me and professional experience pages are created as simple text page components with a few images for increased visual appeal and break up the text. Work sections are vertical stacks of rows of work description, created by mapping over a JSON file, making adding additional projects as simple as adding to the object. The academic experience page contains dropdowns with impactful courses, as well as what was learned in them. The final Build Your Own Computer page contains 11 dropdown modules containing mile high view concepts about computer system design from everywhere in the tech stack.

All dropdowns and cells, including Build Your Own Computer modules are implemented as their own components for two reasons. Firstly, they should control their own state - such as if they are open, what data they are displaying, and what buttons are available. Additionally, making these components allows for significant reusability, especially when mapping over a JSON object. These internal states also only affect the view they are hosted by, so storing them locally requires only the necessary change elements get re-rendered by limiting them to the maximally required scope, and no higher.

The builder modules do use dependency injection to avoid passing state down several levels of the component tree when updating the score, which is updated on the navigation bar once a module is completed.

Some of the more challenging aspects of this endeavor included ensuring reusability of components that are used in several spaces, as well as determining which CSS modifiers yield the expected UI result. For this I used ChatGPT as an aid, not for website design, but as a CSS manual that is easily accessible, often using prompts in the format of “I would like <expected behavior>, what modifiers would help me achieve this”. This helped with learning modifier differences such as padding vs margin, especially when using background/border effects. I also found it helpful for the often unuseful error messages provided due to React errors.

Due to the lack of any back end, this page uses purely CSR for rendering, which is not a problem due to the small number of changes between pages and state updates.