Corey Emery

05-430: Programming Usable Interfaces

November 30, 2018

Homework 8: Final Project Writeup

**Part I**

The purpose of my website is to describe what the wick editor is and provide entry points for users to both the tool itself and the wick editor community more widely (including examples, tutorials, forum chats, etc.). It is meant to provide an engaging, exciting introduction to what can be done using the wick editor, and present a low barrier of entry for users of any skill level.

The wick editor and my website are aimed primarily at elementary and middle schoolers (students who have minimal coding experience, and thus would benefit from the interface of the tool when building their animations).

The information I convey on my site is constrained largely by what the creators want to say about the tool and acknowledgements we must include due to funding we receive from third-party sources. In terms of acknowledgements, this required a section on the “About” page thanking those who contributed to the project (and in the future might warrant a separate “Supporters” page). Looking more at content, we decided to include tutorials and examples to help guide users as they begin using wick editor, images from user-created projects to show the capacity of the tool to create games/animations and inspire users, news updates, and numerous links to external videos, forums, and the editor.

My tool is engaging largely in its simplicity. The design is meant to not clutter the screen so that the user can quickly and easily find what they want. This is critical especially for small children. The use of an elaborate and intricate graphic as the entry point to the site, primary cool colors, and large clickable buttons and links also works perfectly to capture the attention of this age group, make it easy to navigate the site, and promote exploration and creativity.

**Part II**

* **Navigation Bar**  
  Users are able to interact with the navigation bar at the top of every page to move to different pages of the site. Hovering over an element (the word) will cause it to highlight, and clicking on the name of a page that isn’t the current page will render that page with a fade in animation (clicking on create will launch the wick editor, which exists at an external URL).  
  Additionally, when we are at the very top of the page, the background of the navigation bar will not be shown, but after scrolling down this will appear on the screen by fading in from the top. Scrolling back to the top of the page will cause the background to disappear again.
* **Embedded YouTube Videos**  
  By clicking on the embedded videos, users can watch these videos and have access to standard YouTube functionality.
* **Example & Tutorial Cards**  
  On the homepage, there are cards that describe examples and tutorials for the users. A grayed out card means that it is disabled, and thus no interactions will work for these. For active cards, hovering over the card will cause it to highlight, and clicking on it will open a new tab with the appropriate example or tutorial.
* **Learn Page Examples & Tutorials**On the learn page, the examples and tutorials are organized as wide block elements with a down arrow on the right side (suggesting that the element can be expanded). Hovering over these elements will highlight them, and clicking on them will cause content (description, editor launch button, YouTube embed/image) to expand beneath the title element. The arrow will also change to point up (suggesting the element can be collapsed). The same hover interaction exists in this state, and clicking on the wide block will collapse the content.
* **About Page Social Media Icons**  
  At the bottom of the about page, there are five icons representing social media platforms and email. Hovering on these elements will cause the colors to become less opaque. Clicking on the social media icons will link to the wick account on that platform (or the homepage if this account does not exist yet). Clicking on the email icon will open a blank email to wick editor.
* **Buttons**  
  Across the site, there are various reactstrap buttons that will open new tabs to the relevant webpages.

**Part III**

* The main tool I used to construct my webpage was React and Reactstrap (Bootstrap for React). I also used animate.css for select animations across the site, but this was not my primary focus.
* I chose to use React because I wanted to make this webpage have the least amount of redundant code as possible. When working on the shopping cart implementation on a past homework assignment, I was frustrated by having to write large amounts of HTML code in a separate JavaScript file, so the integration of JavaScript and HTML in React resolves this issue. By creating custom components, I am able to create elements that can be reused frequently across the webpage (with different input parameters). Reactstrap offers grid functionality that aids immensely in making the webpage responsive, as well as numerous other features such as cards and buttons that are common in web design.
* I used React (after setting up the basic application) by first giving each page of the website its own JavaScript file and building reusable components and container components from there. Container components are useful in initializing a component’s state (including variables such as whether to display a highlight or expand the content of an element), which can be passed to a presentational component to generate HTML. I used Reactstrap to setup the grid layout of the webpage (making it responsive), and also to include buttons and cards across the site. The various options allow me to use similar components with different parameters for different purposes across the site (i.e. filled vs. outline buttons).
* React helps make my code readable and easier to understand/trace, which is important since this is a project for a client, and will have to maintained in the future by other engineers. I am also able to separate my code into a greater number of JavaScript files rather than one HTML file per page, which makes it easier to maintain and make changes. One only has to go to the appropriate component and make one change rather than search for every occurrence of the same pattern in HTML or find the appropriate function in a vanilla JavaScript file. Reactstrap adds a sense of consistency in style across the webpage, and helps it conform to existing Internet standards and user expectations.

**Part IV**

Much of this webpage is implemented based on the designs provided by my team’s designer—I was given static desktop wireframes and artwork for the header, footer, and page images. I designed the responsiveness of the site during the implementation process, as well as the general color palette (I chose to use blues and greens since these are softer, more inviting colors). I also designed the entire Learn Page, as well as the interactions across the site (highlighting, hover, clicks, etc.). I additionally dealt with the implementation details and how to overcome the limitations of the designs I was provided (i.e. creating padding images for the header and footer to fill the entire screen width).

**Part V**

The biggest challenge I faced was working with designs that I did not create. It is important to experience working in a team environment and interfacing with other members, but on a tight timeframe I was not able to understand every design rationale, which is sometimes needed when I could not implement a design exactly to spec and needed to make some modifications. The second challenge was identifying opportunities for creating new components initially. This is an iterative process that occurs over time after much of the code is already written, which leads to many periods of refactoring and restructuring code that can slow down the development process.

**Appendix: File Organization**

I did not use an assets folder in organizing my code. I had to work within the constraints of the file organization established by create-react-app. Within the src folder are all my files. I organized these by giving each page (and the header and footer) its own folder, all the .scss files its own folder, and all my other components their own folder. Each different component has its own subfolder within this last folder. Files like App.js and Content.js (which don’t fit into any of these categories) are left at the top level of the src folder.