Neha Deshmukh
PUI Fall 2021 - Section E
Homework 8

Implementing a Tool Project
Final Project Pt. 2

Website Links:

Github Repository - https://github.com/nddesh/WebChanger
Live Website - https://webchanger-pui.netlify.app/
Video Demo - https://youtu.be/F3NvAcvVzjI

a) Part 1: Describe your website

My project is an online, visual HTML editor, that can edit websites embedded into the online tool, by scraping the website's original HTML/CSS and modifying it to display changes that a user makes. The website is interactive since users can play around with the design of the external webpages and the tool is responsive for accessibility. Due to security concerns and legal limitations, published websites can only be modified to certain extents and cannot be saved/printed/screenshotted through external JavaScript. This reduced the scope of my project significantly, however, I was able to create styling changes to groups of elements within the website instead of modifying individual HTML elements. To adhere to time constraints and scope of this project, this tool only changes the content for one pre-defined IMDB website, but implementing the same functionality for custom web links is a future possibility.

The intent behind creating this tool was to create a fun, interactive tool for users to visually change web content without having prior knowledge or styling experience with HTML/CSS or JavaScript. This tool is a prototype for an advanced tool that can be useful for web-editing trials, mockup testing, and A/B testing, to try out different styles without having to code/design an entire website. My target audience is a general audience of website users, and possibly an audience of designers and developers who may need this tool for design testing.

b) Part 2: Use a bulleted list to describe how a user would interact with your website.

The tool webpage has 3 main sections: Header explaining what to do, design toolbar to change element styling, and iframes (embedded websites) showing the original and user-changed websites. Once the page is completely loaded with the externally embedded websites, the user can:

- Select changes to the website through dropdowns and color pickers in the design toolbar.
    - The selected styling change is stored in localStorage to be displayed.

- When the 'make changes' button is clicked, the 'Your Design' iframe will reload to display changes made by the user.
    - This modifies localStorage data and displays all changes at once in the iframe.
- Changes are made to the main body of the website, leaving out the menus, headers, and footers.
- These changes can be modified multiple times to play around with styling and test out different combinations of text, colors, and decoration styles.

c) Part 3: Describe what external tool you used (JavaScript library, Web API, animations, or other)

i. Name of tool

I used multiple tools through this project:

- **Node.js** - Back-end JavaScript runtime environment
- **Cheerio** - Tool that parses HTML and XML in Node.js, and provides an API for reading/manipulating the resulting data structure.
- **Axios** - JavaScript library used to make HTTP requests to the browser from node.js, and fetches markup from websites to pass onto Cheerio.
- **Pretty** - npm package that takes HTML strings and structures them in a readable format (beautifies HTML code).
- **Fs** - npm file system to store, access, and manage data on the OS.

ii. Why did you choose to use it?

These packages/libraries are compatible with each other to scrape webpages, by pulling HTML data, displaying it accurately and manipulating it. These tools can only be run through Node.js, since JavaScript cannot directly run them on the browser. To scrape a website and then embed it into my own tool, I had to use Node.js to run Axios, Cheerio, and Pretty. Fs is used to write a local file with pretty HTML to be passed into the iframe.

iii. How did you use it?

To accurately scrape the HTML data of a website, I needed to make an HTTP request (Axios), pull the HTML data from a website (Cheerio), then make it readable to embed into the web browser (Pretty). Cheerio cannot fetch HTML data on its own, so I used Axios to make the HTTP request. Cheerio also does not interpret data like a web browser, so I used Pretty and fs to create an HTML file to display an exact replica of the original website.

iv. What does it add to your website?

Without these packages/libraries, I can only load a webpage into my website through iframes. However, modifying these iframe pages is restricted by the original websites

due to security concerns and privacy laws. Some websites also restrict loading into iframes in the first place using additional security restrictions in their code. Using a local file to temporarily modify websites is one of the few ways I could embed websites to create this tool.

d) Part 4: Describe how you iterated on your HW7 mockups, if at all, including any changes you made to your original design while you were implementing your website.

My hw7 was heavily based on theoretical ideas of manipulating websites using event listeners and functions to manipulate styling. In terms of styling, I changed parts of the layout of the webpage like the header for better legibility, and the layout of iframes to allow users to compare them side by side. I also moved the design toolbar to the center to allow ease of use and visibility.

e) Part 5: What challenges did you experience in implementing your website?

Due to security issues, website and browser limitations, I was not legally allowed to add event listeners to external websites (iframes) so I could not detect clicks inside an iframe. This restricted modifying elements and I had to change elements on the entire webpage instead of only one element. I also wanted to allow users to save their designs by downloading iframes or taking screenshots but the browser does not allow such manipulation through JavaScript, again due to security issues.

Other than that, web scraping took a while to figure out because the given data was difficult to understand without any prior knowledge of how to use Node.js. I also had a hard time loading browser data into my website before I used ‘pretty’ and wrote a local file for iframes.