Sarah Babski
Final Project: Portfolio Website

Project Summary

My project a personal portfolio website. It showcases art, design, and computer science work I have done for school and as personal projects. I organized it as a single-page website with four distinct sections: Intro, About Me, Portfolio, and Contact. Within the Portfolio section, each project triggers further information about it on clicking, without rerouting to a new page. In addition, I used CSS and JS to create transitional elements for interactions, such as fading and scrolling effects. I will also teach myself Sass to help with the CSS, and I will use the Bourbon mixin library and its grid framework Neat to help format my site.

Information to Run

I have uploaded all of my files on a Github repository (https://github.com/sbabski/Portfolio). This includes all of my font and image assets, source code for libraries I used, and my HTML, JS, SCSS, and compiled CSS files. Since it is not hosted online, run it by downloading the files and running it locally. It is best viewed in Chrome, since I found out on the last day that sometimes the horizontal alignment of the parallax groups can be messed up if viewed in Firefox.

Milestones

04-02: I drew up sketches for my site, solidifying my layout. I included all the information I planned on including, the projects I plan on showing, and I annotated it with possible background images and all the interactions that will happen on various parts of the page. I also set up the jQuery library and my basic index.html, style.css, and app.js files (without any real content in them yet).

04-09: I installed Sass, Bourbon, and Neat. I spent a while familiarizing myself with the syntax of Sass and the various ways Bourbon and Neat could help with this specific project. I found some design templates in Bourbon Refills which, when modified, can fit into my site nicely. I used Neat to set up the grid design I will use and styled many of the grid containers, as well as my fonts and some other basic site-wide styling.

4-16: I put in real images and content, as well as some filler content, in my first three sections, and refined the styling to line everything up nicely. I also created a header based off of the centered navigation header used in Refills, and set it up to navigate within the page to each section. I also wrote JavaScript to change the background color based on the

section currently visible, to create a transparent-ish effect without making the navigation text weirdly overlap with the content.

4-23: I added the parallax effect to my first section. This proved more difficult than originally anticipated, since revising my previously-written code to include my new parallax classes messed up my grid format and gave me two scrollbars, as well as sections that vertically overlapped in some places and left gaps elsewhere. I ended up restarting my HTML and SCSS pages, first getting the parallax to render correctly, then adding my styling from my previous grid containers and other classes back on. I also found the header to be annoying, due to how my styling rendered with the header's positioning in relation to my new parallax class. I opted to rewrite the navigation as a menu on the left side, which would not have the same issues with overlapping that I worked on last week due to my grid, getting rid of the now-redundant code from the Refills template.

4-29: I revised the grid format of the work section to be four by two, and added in the hovering transitions, JavaScript for the single-project view, and used the fading effect so that interactions would not be jarring. I adding more fading for when the page initially loads, and found a nice JS library called localScroll to integrate with my menu so that navigation would happen by scrolling animation.

Tutorial

The coolest part of my project is the parallax scrolling, but after looking at several different possible implementation methods, including a few JS libraries, I found a CSS method that was hands-on and better let me understand how parallax views work, and I was able to modify it to be more efficient in Sass and easily do what I was looking for in my site. So I've included both a tutorial for that and for the JS I used in my work section.

Parallax

This parallax method works by assigning one wrapper div to the entire screen, the height of the viewport, and working with relatively-positioned groups for each section within. This allows for scrolling to occur within the parallax class and appear as if each section is positioned absolutely below one another, and it also allowed me to easily create a navigation bar outside of the parallax class that appeared sticky. For my parallax section, I had already broken the content into two sections: the text and the image, so it was easy to add parallax layer classes to each to make them scroll at different speeds. The effect is created by translating each layer a different amount backwards along the z-axis. Normally, this would cause the content that is further back to appear much smaller and would break the rendering of the entire document as one width. However, we scale this multiplying it by a factor that is one more than the translation divided by the perspective we are viewing the page in. This keeps the groups at their regular size while retaining the exaggerated depth. We also have to factor in z-index to each layer class so ones that are supposed to be behind others do not cover them up. Z-index also comes into play with the individual styling of each section. This was not so much an issue for me, since all I needed to do was make sure that my first section did not cover up the subsequent sections. When there are many sections with many layers, however, a wrongly-numbered z-index may render the

entirety of a section, background and all, in front of the entirety of another section. Since the background layers are larger than the base and foreground ones to account for their distance, the illusion that all sections are the same height is broken if the z-index is wrong. Even though only my top section displays the parallax effect, all of the content code needs to be within the parallax code to ensure that they react with each other correctly. Furthermore, I placed each section within a new parallax group so that in the future, if I ever want to add different effects to any individual section, I can easily add them in. But for the time being, the sections past my top one only use the base class, which does not transform them at all.

Work JS

I wanted to contain all the information for my work section within the pre-existing section and within the viewport. To do this, I first organized the work section into two sub-sections: one containing the previews and the other containing the actual work. I decided that within the project previews, I wanted to have hovering toggle the image preview and title, so within the previews section, I split up these two elements for each project. Then, in my JavaScript, I could easily switch between these different views by showing them, hiding them, or fading them in or out. Initially, only the image previews are visible. When hovering over any given project wrapper class, I hide the image and show the title, and when the mouse exits the project, I switch them back. For clicking, I set each project to have a unique id that is used in both the distinct project preview and full project sections. I fade out my view of the previews, search for the id within all of my project options, then write the HTML for the correct project into a separate div and fade it in, along with a close button. Clicking the close button causes both itself and the full project to fade away, and the list of previews returns.