Design Process Documentation

**Setting Goals:**

Even starting this assignment meant a scramble for me, as my usual client (Insight Green) had no purpose at all for any digital experience element, with the domain “insight.green” not even loading any HTML whatsoever and with all of their web hosting devoted to cloud servers for market data storage and processing. That said, the assignment also served as any excellent reason to return to the HTML and CSS I had designed for No Shirt Press, and to add to that a digital experience element that could enhance the user experience or otherwise encourage the user’s engagement. Therefore, I consulted Aron Woldeslassie, the author of the book that the website I designed will promote, and we discussed—in terms, I must note, that were completely devoid of any technical jargon, and perhaps best defined as whatever constitutes a “digital experience element” from the user’s perspective—what types of digital experience elements we could include on the site. In particular, we discussed an analogy between an element of his novel’s narrative (the intrusive thoughts of a character suffering from depression) and the common user frustration with unwanted pop-up ads appearing on a website. Besides that, the novel’s choose-your-own-adventure narrative is also partially written in the second-person voice, which not only predominates the voice used in web-based text content, but does so particularly in web marketing content, of which my own design may stand as an example.

After discussing the above, Aron and I set a goal of including JavaScript on his website for interactive pop-ups within the page window that mimic the intrusive thoughts of the character, with content for these pop-ups that addresses the user directly as its audience, because this experience would indirectly imitate multiple aspects of the book that the website promotes. Aron himself provided me with text content for these interactive pop-ups, which I estimated we could most easily execute using JavaScript.

**Getting Started:**

I must admit, perhaps it’s only after having the experience of previous modules, but I found this module’s more open-ended requirements exciting. I had been hoping to practice (or, rather, to begin learning) JavaScript after having composed my historicized report on an HTML element, and so began the project with intimate knowledge of the site’s design. Having already established not just what type of digital experience element to implement, but also having already created some of the content for that element, I first began searching for examples of raw JavaScript code I could adapt for my own purposes, and saving those within my GitHub repository for this project—surely coded incorrectly and likely in the wrong file-type, but simply for later reference. Still, though I collected some working templates for the JavaScript elements I aim to produce, I nonetheless perceived a great value in working from scratch as much I could, and accordingly downloaded a helpful “Hello World template” for JavaScript from the developers at Google (<https://developers.google.com/closure/templates/docs/helloworld_js>).

I must confess, I enjoyed this project even just in the phase of “getting started.” While the open-ended nature of this assignment seems to account for the lack of a prescribed program or compositional procedure, I needed to first ensure that my system could program in JavaScript. To that end, I checked a StackOverflow query entitled “How to enable jQuery support in Aptana Studio 3” (located at <https://stackoverflow.com/questions/4721124/how-to-enable-jquery-support-in-aptana-studio-3>), but the answers to this query provided conflicting information located on a number of dead links. Therefore, I instead downloaded and installed Eclipse Java Oxygen (requiring me to first locate, download, and install Java Developer Kit 9.0.1), and used the system interface to create a new Java project within my GitHub repository—making sure to create a new branch as I did so—in order to give myself a space in which I could try working from scratch. Besides this, I also discovered a JQuery plugin for Eclipse (downloaded from <https://master.dl.sourceforge.net/project/jquery/OldFiles/ca.ubc.jquery_4.0.3b.zip>), which I added to my Eclipse program files so that I might have the option of building a JQuery factbase.

At that point, I felt completely confident that I had the tools necessary to create the digital experience element that Aron and I had previously discussed. However, the hurdle yet remained of my total inexperience in writing JavaScript, let alone implementing it within a currently-static webpage. Thus, I began using Eclipse Java Oxygen to experiment with JavaScript in creating “helloworld\_js” as a new project in Eclipse (complete with another GitHub branch) according to the steps Google developers provided. Not knowing where I might find “the executable java on your path” through which Google’s instructions require me to run a command, I created an external tool prompt inside of Eclipse to run a command “with the executable java on your path.” After realizing that the “path” mentioned in Google’s instructions implied that I should use the .jar file in the “helloworld\_js” project. Still, trying to run this external tool resulted in the notification that an exception occurred executing the command line, with my computer unable to run the .jar program included “helloworld\_js” because “CreateProcess error=193, %1 is not a valid Win32 application.” I felt an impulse to investigate this further, on the intuition that install a 32-bit version of JDK 9 would resolve the issue, but soon discovered that Oracle has not released a 32-bit version. When I simply tried to open the .jar file within Eclipse, a prompt appeared informing me of “Editors available on the Marketplace.”

Feeling a growing sense of desperation at this point over the concern that I might need to find an alternative method for composing my JavaScript, I chose to install Bndtools 3.0.5: the third editor in the list, but the one with a description best matching my needs. With this final installation, I successfully executed the .jar file and viewed it in Eclipse. However, I’m still finding no success following Google’s “Hello World” instructions, and can find no means of executing the specified command. Knowing that I’m working under a deadline, I turn instead to the reliable source of Lynda.com, where I viewed “JavaScript Essential Training,” while also downloading the provided exercise files and adding them to my GitHub repository in a fresh branch. I felt quite foolish early on, as “JavaScript Essential Training” immediately suggests the Atom text editor, a program with which I already possess some hands-on experience. Per the recommendation of the video, I installed Atom Live Server, downloading the zipped folder from the program’s GitHub repository and manually added the contents as a plugin to my own program files.

Clearly, Lynda.com is where I should have started all along. After learning the basics of JavaScript in the first two chapters, I moved on to chapter four, with the impression that a series on “Functions and Objects” in JavaScript would likely serve my purposes. As deadlines approached, I became less interested in the scope of JavaScript functions, and decided instead to search w3schools.com for a template I could modify to suit my purposes (which I found at <https://www.w3schools.com/howto/tryit.asp?filename=tryhow_js_popup>). While I found some suitable alternatives to what my client and I had planned, I personally considered them somewhat amateurish, and hoped to find some means of designing something more engaging in my increasingly limited time.

**Process:**

Upon finally beginning to compose JavaScript, I decided to err on the side of caution, and to tag my scripts as deferred. Whatever time I lost in trying to acquaint myself with other software for composing JavaScript, I cannot say with certainty that I have yet succeeded, as my JavaScript elements appear on some browsers and not on others. The time constraints my explorations imposed on me in producing this draft also forced me to use the Opentip CSS (<https://github.com/enyo/opentip>), as I became increasingly desperate to see at least some of the content that I had planned with my client on designing to finally appear. I have attempted three different methods of using JavaScript to add an interactive element to a user’s solicited decision to buy or not to buy the book, some in isolation and some in concert, and have not yet found a consistent method or combination of methods. Additionally, I suspect that Opentip’s CSS conflicts with that I had previously designed for the presentation of this website, although I admit this may also result from a poor execution of JavaScript in the HTML.

Following my submission of an unusable draft of my project for peer review, I began to consider new approaches for implementing the features my clients and I discussed. After personally consulting the programmer of another client, I discovered that I could implement these features via HTML and CSS as “tooltips,” with the same programmer offering a JSFiddle (<http://jsfiddle.net/technotarek/2htZe/>) and a repository of CSS with which to customize my tooltips (<https://github.com/gw19/eleganttips.css>), I developed some fresh approaches. Since the three-column division in the body constituted one feature of my peer-review submission that had made the website unusable, I decided to eliminate the code for the features from this index’s HTML and CSS, now viewing the project with the goal of producing simple examples of the features my client desires that we can later apply to the final website.

Fortunately, I succeeded in modifying the JSFiddle my friend provided so that I could provide a functional example (though a minimal one) of the client’s desired interactivity features. However, I ran into my next obstacle in trying to implement one of the client’s specifications for these features: namely, that from among the general pool of tooltip content my client provided, he requested that specific pair of tooltips should appear attached to each component of a binary choice offered in text to the website user. In other words, after having eliminated my three-column structure (at the time a difficult task to produce) to produce a working example of the interactive elements we had planned, I now faced the challenge of producing a two-column structure for the sake of presenting the binary choice to which I could attach the client-specified pair of tooltips. Eventually, however, I abandoned even this two-column approach, and—even while writing some CSS and HTML which I feel confident the final website will include—narrowed my approach entirely to producing a website draft containing example tooltips.

Throughout the process, I eliminated much of the presentation styling in HTML and CSS as it had existed for the original website. Many redundant artifacts of code still exist within the resulting files, I’m sure, but nonetheless, I finally had a “playground I could rely on” for practicing how to implement tooltips into HTML and CSS, and for customizing those tooltips as best I can.

**Results:**

Implementing the tooltips that a friend I consulted had provided to me proved an easy endeavor, as did customizing the color and content of the tooltips. Although I still believe I could have spent more time experimenting with HTML and CSS to produce tooltips better customized for the purposes of the website (such as setting locations on the page in which various tooltips would appear), finally I had arrived at a draft containing interactive features for which I would feel comfortable conducting a usability test, and didn’t want to risk customizing those tooltips any further, lest my attempts render the website once again unusable.