**CS 319 Project 2**

*Group 104*

Instructions for running code: This project is a chrome extension done in HTML and JQuery. The JQuery file is already imported into the local files of the project, so it only needs to be run through the exntensions of a chrome browser.

Kenneth Black, Matthew Vanderwerf

Spring 2017

**CS 319 Project 2**

*Google Chrome Extension*

Checklist

* Use of APIs (activeTab.js 10-58)
  + Ajax calls
  + URL call to get IP address (11-19)
  + Use of Teleport API for location (20-42)
  + Use of Weather API for weather information (43-58)
* Settings (settings.html + setting.js)
  + Theme.css to hold themes for extension
    - Save theme cookie in settings
    - Access again inside of main.html
  + Dropdown button with multiple themes (settings.js 38-81)
    - Dropdown.js
  + Hide Location or Weather Information (72-93)
  + Save settings and back button
    - Use of .png files as buttons
    - Back button to cancel and save button to apply setting to main page
* Note Taker
  + Ability to create new notes (activeTab.js 67-80)
  + Access previous notes in note history (activeTab.js + Note.html)
  + Different html page to edit note title and body (Note.html and Note.js)
  + Delete notes through settings (settings.js 94-99)
* Imports
  + Manifest.json file for chrome extension (manifest.json)
  + Import Jquery.min.js
  + Jquery cookie import from github to save background information (jquery.cookie.js)

Group 104

Kenneth Black

Matthew Vanderwerf

Google chrome customizable popup extension with information pertaining to current location as well as a note saver for html pages.

Table of Contents

1 …………………………………………………………………………………………………………………… Quick Explanation

2 …………………………………………………………………………………………………………………………………. Checklist

3 ……………………………………………………………………………………………………………………………….. Title Page

4 …………………………………………………………………………………………………………………. TOC (Current Page)

5 …..……………………………………………………………………………………………………………………………. Overview

5 …………………………………………………………………………………………………………………… New and Complex

6 .……………………………………………………………………………….. Bloom’s Taxonomy (APIs and AJAX calls)

7 .…………………………………………………………………………………………… Bloom’s Taxonomy (User Events)

8 .…………………………………………………………………………………………………… Bloom’s Taxonomy (Jquery)

Overview:

Our creation for portfolio two was a chrome extension. We made a customizable popup screen, that gives basic information such as your location and weather information. This was done through a few different APIs and URLs, using your IP address to find your location and the remaining information.

The extension also has a note saver that saves notes taken. The note is saved in the notes taken section with the note title as the link to the note. Once clicked, it will redirect you to the concurrent note and give you the ability to edit the note title and body. The main page of the extension can have its css properties altered using the settings page. The settings page allows every component of the extension to be shown or hidden, as well as allowing different preset themes. There are 5 preset themes that the chrome extension can be customized to. You can also delete all the notes from the settings page. When reopening your note, a new html page is opened.

New and Complex vs. Non-Trivial Tasks:

Some of the easier things were styling the css elements and laying out the html pages. But JQuery was still somewhat foreign to us, and we learned a lot in the process of creating this web extension. Some of the style changes we’re difficult to pick up at first (like positioning), but once I got the hang of it, laying out the html pages was simple. A lot of things we’re extremely complicated at first, until I found a proper way to do them, such as cookies and css themes. I was trying to save the data inside of the tags for the longest time to transfer css changes. Jquery does not have a built-in cookie functionality, and I wasn’t even thinking of using cookies at the time. But cookies are the quickest way to send quick collections of data, without using a php script and access to a server (I did not want to create entire server space for such simple data across html pages).

Things started to get more complicated when I had to get css changes from one html page to transfer to another page, calling the apis, and creating the separate Note.html page. It was more difficult to work with the Ajax calls than we thought it would be. The urls each sent data differently, so converting and formatting the json was frustrating at first. One would send data as a json string and the others sent the data as a json object, so we had to learn the difference between JSON.parse and JSON.stringify to split up the JSON data into readable text. For convenience with JSON, I used public APIS that didn’t need keys or cvs for easier access. It would have been ridiculous to use a fancy API for what I was trying to create.

One of the more difficult parts was getting a variable out of an ajax call and into a different function. I needed to take the IP Address from one API and use it to find the city location using a different API, and I did this by making sure the functions were not done asynchronously like they were doing by default. The note saver was much more simple to create on the main.html. But to get those notes to save across html pages, with the correct note title and body to edit, took a lot more effort with the cookies than anticipated.

Bloom’s Taxonomy

Even though my partner and I had never developed a chrome extension before, our experience in class with web development gave us some insight on how to start such a project. HTML was used for the UI, and since we were making url calls and a lot off css changes, we figured JQuery would be a more suitable language than Javascript as the scripting for this endeavor.

Our initial idea was to make a website blocker to block sites that a user found distracting, but this was already a common extension on chrome. We also realized this could be done with some simple GET and POST requests. A popup screen for quick notes while reading through a web pages would be a useful utility to use as an extention. We added weather and location to give the user information that may reap hidden benefits for their day and they may not have searched for otherwise such as the skyline and weather.

This project was a great way to learn APIS as well as an in-depth application of JQuery. I had only implemented APIs in python, so the use of APIs in web development was new to me. Altogether, a chrome extension was a great way to learn many aspects of web development. I had already implemented the Teleport API for a previous job to find a user’s location and the average salary of the area using their IP Address. So, using this API wasn’t too unfamiliar. But I had done this in python originally, so doing the same in JQuery took a little while to configure.

At first I had a problem with the ajax calls and formatting the JSON data. I couldn’t access the specific data inside of the JSON and I was accessing the entire object. Once I learned how to format the data with JSON.parse and JSON.stringify, I could access the JSON data separately. After this, the problem became taking the variable being grabbed from one ajax call into another ajax call url. We did this so that we could take the IP Address from one API, and then use it to find the user’s location in a different API. We then used that data to get the weather information for that location. Global variables wouldn’t do the trick, and I struggled on this for a while. I eventually realized the AJAX calls were being done asynchronously, so we had to make sure the calls were done in order so data can be stored into a variable that the next AJAX call can access. We just set the async variable to false.

I wish I had gotten the JSON data down earlier, so I may have implemented more APIs into the project. I would have liked to add a stocks API as well. It was also just step that took me way too much time compared to much more difficult steps that I figured out more quickly. I definitely learned a ton by finding out how to synchronize the AJAX calls, and I’ll be able to use this for my web development internship this summer.

A note gets added by interacting with the add button on the right. Once clicked, two text fields will appear as input for the note title and body. At first I had just a note body, but realized a title would be easier for accessing a history of past notes. I also was going to limit the length of the note, but when editing there is a lot of space in the Note.html, so I figured the notes could be any given length the user wants. The note can only be added with both text fields filled and can be cancelled at any time. When the note is added, the title of the note becomes a clickable link in the note history on the bottom of the main page.

When the note is being added, it will pop up on the main screen. But when I started coding for the editing of a note, I wanted to try creating a different html page for the note to be added on (even though I knew this was substantially more difficult and a major design flaw I wanted to see how I could transfer data from one html page to another). It would have been so \*\*\*\*\*\*\* easy to have a click function on the note history links that unhide the div section in lines 68-73 of the popup.html and set the values of the text fields as the previous note title and note body. But noooo, I said screw that and decided to do a whole additional html page to challenge myself and then entered the world of cookies.

Unfortunate thing about jQuery, is there is no built-in functionalities with cookies. So, I had to pull a repository called jQuery-cookie by carhartl off GitHub that I included in the project files. There was a more recently updated cookie extension for javascript, but using Jquery I had to stick with this one. I’d never worked with cookies before, and I was floored by how much time it saved transferring data between pages and every interaction with web pages and the browser in general. Cookies were an awesome life saver that I wish I had utilized earlier in my project, because it would have saved me tons of time and allowed me to structure something a little more powerful too. But it still was extremely tedious saving and accessing all the cookies with loops through two different html pages, when I could have stored the notes locally within my main page in like 10 seconds. But it was a learning experience for me (and one of the most frustrating parts of the project).

The note.html comes up with a note title and note body and the two buttons found in the settings.html: back and save. The note can be edited in this new html page and can either be saved or cancelled by hitting the back button. At first the note history used a tags as the reference to the note.html. But the note ids would not be referenced by .click for an a tag, so I used the regular p tag instead. I had an input button that could be referenced by a .click function, so I could save the note title and body in their placeholders for the Note.html.

I couldn’t decide on whether to use Javascript, JQuery, or PHP. When we realized PHP was more server side, I decided to stick with JQuery or JS, seeing as this would be focused on the front end. It didn’t really matter whether JQuery or JS was used for scripting, but JQuery seemed easier to manipulate css elements and call APIS. Also we had less experience with JQuery so there was more to learn with using this language.

For the settings, we had two separate categories we used for the css changes. One thing I did was a dropdown box to change the theme of the chrome extension. The themes were stored in the theme.css file, so they can be preset and called from both html pages. I grabbed a dropdown menu script online and then used that for the settings interface button. Only problem with the script was in Javascript, so I had to convert it to Jquery. Because of this I couldn’t implement the function to get the dropdown bar to minimize when clicking outside of the screen. But every other aspect of the dropdown button works fine. When clicked, five buttons with exquisite names, each with their own respective background colors and font, drop down from the menu. When one of these is click, the dropdown menu bar will change to the respective theme clicked. The theme will also be applied to the settings screen to preview what it will look like on the main screen.

The settings also have checkboxes to hide the location and/or the weather. Checking either and saving the setting will hide that given body of text on the main extension. You can also click a button to delete all the notes from your note history. The boxes were initially inputs of the checkbox type, but it was a far more difficult to grab information on whether the box is checked or not, because there is no specific code to check this in Jquery. So I just created a simple 10px by 10px box where the background color of the box is what decides whether the option is checked or not. This was much easier to write basic jqeury if statements for.

Hitting the back arrow will cancel all changes made to the settings page and go back to the main page. I used png files for the buttons to make them clearer icons. Hitting save will store the css changes into cookies and redirect you to the main page, where the css changes will be applied from the cookie. This is so a user can see the changes applied to the settings page before saving them to the main screen. I would have added individual note deleting and a stocks API, but we achieved what we set out to do.