

ExplainShell for Chrome

Progress Report

Submitted to

Thomas M. Keating
Assistant Teaching Professor
School of Computer Science
Carnegie Mellon University

Prepared by

Justin Gallagher
Ted Li
Jacob Zimmerman
Howard Chen

School of Computer Science
Carnegie Mellon University
November 14, 2014

(Professor Keating granted us a two-day extension)

Contents

1	Overview	1
1.1	Purpose	1
1.2	Project	1
2	Literature Review	1
2.1	Chrome extension reference	1
2.2	Setting Up a Node.js Stack with Express & MongoDB	1
3	Progress	2
3.1	General Assessment	2
3.2	Status	2
3.3	Projections	3
4	Recommendations	3
5	Discussion	3

1 Overview

1.1 Purpose

This report intends to inform the client of our current progress in building ExplainShell for Chrome. Specifically, we will cover additional information that we have found relevant to our project, the goals we have accomplished, what we still need to complete, required changes to our original procedure, and our plan for finishing the project at the deadline. We will conclude with a discussion of what exactly we expect to produce when finished.

1.2 Project

ExplainShell for Chrome is intended to help the user understand Unix shell commands found online. With the aid of our Chrome extension, users will be able to click on commands and be directed to the relevant `explainshell.com` page. Our application eliminates the need to copy and paste the command into `explainshell.com` and places a friendly reminder of ExplainShell's utility in the reader's view. Combined, these two aspects eliminate the largest barriers to people using `explainshell.com` freely, thus promoting its use.

As a complement to this utility, each click that users generate by following links to `explainshell.com` from our Chrome extension will be sent off anonymously to our ExplainShell Trends site, which will compile a database of clicks which we will be able to query for statistics such as most recently clicked commands, top referring sites and pages, most popular commands, and more. Through this site, ExplainShell for Chrome users *and others* will be able to learn about new, helpful shell commands.

Thus, this system will both help bring attention to commands that users don't understand as well as introduce them to new content generated by their peers. It solves issues related to malicious code execution (because potential victims will be more wary about running unknown commands), Unix shell education, and community involvement.

2 Literature Review

We have found a number of reference materials useful along the development process, mostly relating to the actual implementation of Chrome extensions and web apps in Node.js. Of the sites we've visited, those that we document here have been the most crucial to our current progress.

2.1 Chrome extension reference

2.2 Setting Up a Node.js Stack with Express & MongoDB

<http://cwbuecheler.com/web/tutorials/2013/node-express-mongo/>

3 Progress

We now present a general assessment of our progress so far. We will also propose potential changes to our original schedule.

3.1 General Assessment

Up until this week, we have completed the development of Phase 1, 2, and 4. So far, we have built a Chrome extension that parses and searches any web page to identify any shell commands. As the extension finds any potential commands using a set of heuristics, it will turn the original command text into a clickable link, which will redirect the user to the corresponding usage documentation at the ExplainShell website.

We have also completed the development of a back-end server that tracks the commands that users search for using our extension. The server compiles a database of clicks, which we will use to present various analytics, including most recently visited commands, trending commands and top referring sites.

The next step is to complete Phase 3, during which we will integrate the front-end interface with the back-end server. To do this, we will likely need to contribute to the existing ExplainShell code base to add in relevant APIs.

3.2 Status

Below is a general list of tasks that we have completed so far. Note that the items listed below correspond to our original Gantt chart we included in our initial proposal.

- Create initial Chrome extension
- Test initial Chrome extension
- Parse page, add in external link
- Create in-page popup on hover (partially completed, see below for details)
- Setup basic analytics / trends site
- Add analytics support to back-end

Note that we have only partially completed the development of the in-page popup. Our original proposal was to create a custom-styled popup to display relevant information. This however would require us to first contribute to the ExplainShell API, since we need to add in relevant API that allows us to retrieve relevant reference documents from ExplainShell through web requests. Therefore, the development of the in-page popup cannot be fully completed before the ExplainShell API is available.

Therefore as of now, we have to temporarily compromise with an in-page popup that uses an `iframe` web element to display the reference. This falls short to our original proposal, since an `iframe` web

element does not allow us to customize the look and feel of the popup, nor would usually suffer from a slower loading time.

As a reference of the work we have completed so far, please refer to the link below for a look at our Github repository. (<https://github.com/justingallagher/explainshell-for-chrome>)

3.3 Projections

Our project is slightly falling behind. This is partly due to the fact that the time cycle of contributing to the ExplainShell open-source took longer than we thought. It requires the approval of our pull request by ExplainShell code reviewers and the integration of our code into the ExplainShell code base. Since part of our project is dependent upon the integration of our code into the ExplainShell code base, we may have to make a few adjustments to our original schedule to deliver our final product on time.

To this end, we will need to make some adjustments to our original plan. Particularly, if we cannot contribute to the ExplainShell code base by our project timeline, we may need to compromise with our existing `iframe` solution for the in-page popup, instead of using a custom-styled popup as proposed originally.

Our next major step would be to integrate the back-end logic with the front-end. After the integration, our back-end server would be able to track what users are searching for and provide relevant analytics. This also would be the final portion of our project. We expect it to take one to two weeks to complete.

4 Recommendations

Drop the popup (or reschedule to after done debugging Phase 2)

5 Discussion