Overcoming Technological Barriers to Accessing Ancient Greek Philosophy

An Honors Thesis

By Bradley Hook

Advisor Jeffrey Rydberg-Cox, Ph.D.

Background

In a world where we commonly hold more raw information in the palm of our hand than is contained in all of the printed books in the United States Library of Congress, it seems odd that some information is so difficult to access and use. The ideas conveyed in the Classical works of the ancient Greek philosophers serve as the foundation to much of western civilization, yet it seems the overwhelming majority of contemporary westerners have had little, if any, exposure to these important texts. Technologies like DuoLingo have been developed to make contemporary human languages readily available to every person with a smartphone, but the works of Plato and Aristotle seem to require traditional methods of years of rigorous study to even begin reading these works in their native language. Based on this observation, the question arises as to whether there exists some technological barrier to easily access ancient Greek philosophy texts in the original language.

That question led to the development of a non-traditional honors thesis project. The goal of the project would be to implement features of existing technologies in a manner which better facilitates access to ancient Greek philosophy texts for average philosophy students. If technical barriers did exist, they would be encountered and documented during the development process. In addition, existing solutions would be researched and evaluated against the problems under consideration. With the primary result being computer code rather than a traditional research essay, a method for reviewing and publishing the completed project was needed.

The thesis concept was jointly approved by the faculty of the university's philosophy department as well as the classics and ancient studies program. A faculty member proficient in both ancient Greek and numerous computer programming languages was selected to serve as the

advisor on the project. An appropriate advisor was critical to ensure that useful guidance could be offered and that a meaningful evaluation of the code could be completed.

Evaluating Existing Solutions

A review of existing solutions was the first stage of research. It was quickly realized that there were numerous existing projects aiming to develop solutions for accessing ancient Greek. Each solution had its own advantages and quirks, and most served some specialized purpose. These tools work well within the confines of a given circumstance, but in pursuit of the thesis project, some potential barrier must be identified and analyzed.

One set of commonalities among the tools considered below is the trade-off between two different kinds of *portability*. Web based tools provide excellent cross-platform portability, but they cannot be pulled onto the local device for use when reliable connectivity is not available, limiting the physical portability to locations where connections are available. Stand alone applications remain completely functional without a connection, making them portable in the sense that they can be taken anywhere, but they suffer from poor portability between computing platforms. With this seemingly apparent barrier in mind, four common tools used for studying ancient Greek were considered during this project.

Perseus Hopper

The mission of the Perseus Digital Library is “to make the full record of humanity - linguistic sources, physical artifacts, historical spaces - as intellectually accessible as possible to every human being, regardless of linguistic or cultural background” (“Research”). At the present time, the contents of the Perseus project are primarily taken from Greco-Roman sources and include classical Greek, aligning it perfectly with the aims of this thesis project.

A major part of the Perseus Digital Library endeavour is embodied in the Perseus Hopper, an open source database and web interface for accessing a variety of texts. There are over thirteen million words in the ancient Greek collection held within the active Perseus Hopper ("Greek").

The Perseus Hopper is a shining example of excellent technology enabling access to ancient texts, and every known ancient Greek philosophy text seems to be contained in the collection. However, the Perseus Hopper is not suited to all uses. It is an excellent tool for a student focusing on classical and ancient studies, but for a philosophy student who is seeking targeted information, the Hopper can return overwhelming amounts of data.

The Perseus database contains Greek texts from a period of about one thousand years, and any natural language would evolved considerably during such a long time span. Consequently, when the Hopper returns the definition for a common word, the result can contain several pages of possible meanings and hundreds of citations for context. An excellent example is the word λόγος which returns five printed pages of definitions and citations. This makes sense given that Perseus is designed to enable users to read texts in a linear fashion rather than serving as a searchable lexicon.

Because the Perseus Hopper is an open source project, the computer code and related data files can be used in whole or in part to create new instances of the Hopper, to extend the existing functionality, or to build new projects. The design of the Hopper requires a considerable amount of server software to be setup before the Hopper can function. As a result, even though the Hopper could be duplicated on some local systems for offline use, the potential platforms that this will work on is limited to those which are capable of running the server dependencies. This

makes it impractical, if not impossible, to run an offline instance of the Hopper on mobile platforms such as Android or iOS.

Anki

While Anki is not specifically related to ancient Greek or languages, it is an incredibly useful general purpose electronic flashcard system. Users are able to build and share decks of virtual flashcards on any topic. Recent versions of Anki support accessing decks via a web browser, making it much more portable and dynamic. Anki is also open source, which has allowed other developers to create derivatives that run on most common platforms.

One such derivative is AnkiDroid, a version of Anki that runs on the Android platform. Android is one of the most used mobile computing platforms in the world, with over one billion unique active devices as of June 2014 (Pichai et al.). Unfortunately, Android does not include any fonts which have glyphs for ancient Greek accented characters, so AnkiDroid failed to provide a useful ancient Greek study tool out-of-the-box. Thanks to AnkiDroid's open design, the AnkiDroid Fonts extension was created to make it easy for users to install a unicode font supporting not only ancient Greek, but also a wide range of other languages.

Anki and its derivatives showcase an application model where the developers create a framework and users fill in and share the content in creative ways. This design makes Anki one of the most generic and useful study tools available. Anki comes close to solving the proposed barrier, because it works on virtually every operating system via its web interface, and it can be used offline on most operating systems, including on mobile devices. However, Anki does not achieve this as a single application or project. Anki, AnkiWeb, AnkiDroid, and AnkiMobile are four separate applications that share a (mostly) common file format. While this arrangement

results in a functional platform for Anki, it actually highlights the portability problem addressed in this thesis project.

Suda On Line

The Suda (Σοῦδα) is a medieval encyclopedic lexicon with about thirty thousand entries. Suda On Line is a project that aims to put the entire Suda, including translations and annotations, on the Internet ("Welcome"). Much like the Perseus Hopper, Suda On Line is a tool aimed at academicians focusing on classical antiquity as opposed to students who may be focusing on contemporary topics which reference or derive from ancient texts.

A search for the word λόγος in the Suda On Line database returns about five times more printed material than the Perseus Hopper. The lexical entry for λόγος is only about one printed page, but the database returns all material from the collection that uses the word or any of its derivatives.

The Suda On Line project adheres to an open model by offering all of the content under a Creative Commons license. This license permits the content to be freely shared and reused so long as derivatives are openly shared in the same manner. However, the source code for the engine which processes and delivers the content does not appear to be available to the general public for duplication, analysis, or extension. This makes the Suda On Line project strictly bound to devices with a functional Internet connection.

Alpheios

In an effort to improve web based access to ancient texts, the Alpheios project has developed an extension for the Firefox web browser. Alpheios provides a rich feature set, including the ability to easily lookup any ancient Greek word on any webpage. In addition,

Alpheios allows users to build custom vocabulary lists, explore the morphology of words, access grammars, and take quizzes.

The Alpheios extension is the core of the whole project. As an extension to Firefox, Alpheios can run on Windows, Mac, or Linux. Unfortunately, in the rapidly changing world of technology, this serves to limit Alpheios. As previously mentioned, Android is widely deployed around the globe as a major platform. There is a version of Firefox which runs on Android (called Firefox Mobile), but Alpheios was not designed to run with this version of Firefox. Other mobile platforms, such as iOS, are left out in the cold ("Firefox"). In addition, the top ten laptops sold on Amazon in June 2014 were all running an operating system where Firefox is not available (Pichai et al.).

The Alpheios project likely established itself as a Firefox-specific browser add-on because Firefox was the dominant open source web browser at the time. Alpheios is open source, so the code can be easily adapted into other similar projects. The open nature of Alpheios has also led to collaborative efforts between Alpheios and the Perseus Digital Library (Bamman 10).

Selecting Possible Targets

The old tools were built around traditional application architectures necessary to overcome the hardware limitations of preceding generations of technology. The Perseus Hopper and Suda On Line rely on a server-client architecture where the processing is done on a dedicated server and information is handed back to the client device for display to the user. Anki and Alpheios rely on native compiled code to deliver high performance on compatible platforms.

While these models do work well in many scenarios, evolving technology may allow for different approaches. The mobile phones of today have more processing power than the servers from a decade ago. Advances in Just In Time compiling and Interpreted code engines have made

it possible for robust applications to run on a wide range of devices without suffering from huge performance penalties (Pichai et al.).

The target for development should be an application that leverages these two points. The entire application should be able to run entirely on the client device, regardless of whether it is a desktop computer or a mobile phone. Beyond that, it should use a standardized computer language that is designed to be interpreted on-the-fly from the widest possible range of existing platforms. These design goals suggest that an entirely self-contained HTML 5 application using ECMAScript (JavaScript) may be a viable target.

A quick-search vocabulary framework fits the design goal. The complexity of this sort of application is reasonable for the scope of the thesis project, and the size of a typical vocabulary data set is sufficient to test the limitations of the implementation.

Evaluating Existing Limitations

Using ECMAScript to handle large datasets and complex operations has historically been problematic. As the world moves towards cloud based applications, the platforms are evolving to compensate, and as a result the problems are dissipating.

In 2007, a small team of developers began writing code for an analysis tool to deal with large sets of financial data[1]. HTML and ECMAScript were selected for the initial platform, and development began. The platform was changed relatively quickly after the developers discovered that a high-performance workstation suffered multi-second delays when processing a data grid with 10,000 cells. An empty grid could cause the system to lock-up for five to ten seconds from a single click of the mouse. The final solution was developed using the more traditional client-server model.

[1] This information comes from personal experience working on the project.

Around the same time, developers at Google created a benchmarking tool to measure the performance of Javascript in web browsers. The tool was designed to produce a reference score of 100 on the then-current workstation hardware which was running Firefox 2 (Bak). The benchmark was designed so that as performance improved, the scores would get proportionately higher. Today, in 2014, an average computer produces a benchmark score of about 17,000. These performance gains allow current applications like Google Spreadsheets to load 26,000 cells in a default empty spreadsheet with no noticeable performance problems even on modest hardware.

In addition to historical problems with handling large datasets in active processing, ECMAScript has been plagued with a complete lack of storage options. In his digital book *Dive Into HTML5,* Mark Pilgrim traces the history of ECMAScript storage issues back to the earliest days of the World Wide Web. Early storage allowances barely permit enough room to store a simple font to display ancient Greek characters, let alone storing the text content or script libraries needed to present a functional application.

Numerous proposals for robust storage mechanisms have been proposed over the years, and the main *File API* standard is still only classified as a draft ("All Standards"). Fortunately, some developers have created highly portable script libraries which are able to implement some variant of the draft standards on almost every current platform, so it is now possible to store significant amounts of data through an ECMAScript application.

Implementation

The implementation created for this thesis is built on HTML5 technologies. To avoid reinventing the wheel and to demonstrate the techniques commonly used in current applications, several open source components have been embedded into the project and appropriate credits

and links can be found in the source code. To ensure that the code can be easily copied to any device for offline use, the entire project has been compiled into a single HTML file. To design a truly portable and self-contained system, all vocabulary data is added directly to the HTML structure during use, and then the contents of the HTML file can be easily stored for future use.

How to Retrieve the Code

As an open source project, the code for this thesis project is stored in GitHub. It can be retrieved using normal GitHub procedures (see https://help.github.com). The code base has been titled "HTMLVocab" and the repository itself is available at:

https://github.com/bdhook/HTMLVocab

Usage

The application itself is an empty framework which can accommodate any ancient Greek vocabulary list. When loaded in a modern web browser, the application will present a title bar, a *Tools* menu, a row of buttons representing each letter of the Greek alphabet, and a text box for searching. To use the tool, vocabulary words must be added to the application. This can be done from the *Tools* menu.

The first option is to append one word at a time to the vocabulary using the appropriately named menu item. The dialog to append a word provides example text and asks for four pieces of information. This method is designed for manually building or modifying vocabulary lists in an ad hoc manner.

A second option is to append multiple words. The dialog for this option accepts a tab-delimited set of vocabulary words. Each line should contain the values required to add a single word. This method is designed to populate a large set of vocabulary words into the framework.

The last option in the *Tools* menu is the *Save Current List* item. This option takes the entire contents of the browser window and presents it as a file for the user to save. The user can then choose to save over the existing file, or to create a new copy containing the current changes.

Once words have been populated into the list, the text box can be used to filter the list of words. Direct entry is accepted if the user has the ability to type Greek characters, but the buttons for each Greek letter can also be used where direct entry is not available (such as on a mobile device).

Each entry in the vocab list can be expanded or collapsed by clicking on the heading. From the expanded view, an entry can be edited by clicking the appropriate link. The edit dialog works exactly like the *Append One Word* dialog.

All features have been designed to work on a variety of platforms. The interface has been tested on desktop computers, tablets, and mobile phones. The application should work on any machine capable of running a current browser as of this writing, though hardware more than two years old may have degraded performance depending on the hardware capabilities.

Going Forward

The code implemented in this thesis demonstrates that a modern ECMAScript application could potentially handle a highly sophisticated application that is portable both in terms of platform and available connectivity. By optimizing data processing through embedded relational database engines that are now available, significantly larger and more complex data sets could be processed directly in the browser.

There is a trend towards low cost end-user computing devices that rely on cloud-based server to power robust applications. However, even these low cost devices are more than capable of running the application developed in this thesis.

Based on the application designed and tested during this thesis project, it does not appear that there are significant limitations in hardware or software frameworks that would inhibit the creation of significantly better tools for ancient Greek, or other languages. With the source code of many of the existing projects, as well as the code from this project, being made publicly available under open source licenses, the world will hopefully see rapid advancements in available language tools over the next few years.

Works Cited

"All Standards and Drafts: Javascript APIs." W3C. World Wide Web Consortium, n.d. Web. 30 June 2014. <http://www.w3.org/TR/#tr_Javascript_APIs>.

Bak, Lars. "Google Chrome's Need for Speed." Chromium Blog. Google, 2 Sept. 2008. Web. 30 June 2014.

Bamman, David, and Gregory Crane. "The Ancient Greek and Latin Dependency Treebanks." 10 Dec. 2010. Web. 30 June 2014. <http://nlp.perseus.tufts.edu/docs/latech.pdf>.

"Firefox for Android." Mozilla Support. Mozilla Foundation, n.d. Web. 30 June 2014. <https://support.mozilla.org/en-US/kb/is-firefox-available-iphone-or-ipad>.

"Greek and Roman Materials." *Perseus Digital Library*. Tufts University, n.d. Web. 30 June 2014. <http://www.perseus.tufts.edu/hopper/collection?collection=Perseus:collection:Greco-Roman>.

Pichai, Sundar, Matias Duarte, Dave Burke, Avni Shah, David Singleton, Patrick Brady, Rishi Chandra, Urs Hölzle, Greg DeMichillie, Eric Schmidt, and Ellie Powers. "Keynote Presentation." Google I/O 2014. Moscone West Convention Center, San Francisco. 25 June 2014. Speech.

Pilgrim, Mark. *Dive Into HTML5*. (2011): n.p. Web. 30 June 2014.

"Research." *Perseus Digital Library*. Tufts University, n.d. Web. 30 June 2014. <http://www.perseus.tufts.edu/hopper/research>.

"Welcome to the Suda On Line." *Suda On Line: Byzantine Lexicography*. Stoa Consortium, n.d. Web. 30 June 2014.