Web of Tabs: Browsing Internet Information Intuitively

Luke Moryl
California Institute of Technology
elmoryl@caltech.edu

Daniel Obenshain
California Institue of Technology
dobensha@caltech.edu

Walter Mostowy
California Institute of Technology
mostowy@caltech.edu

ABSTRACT
The introduction of tabbing for Internet browsing has caused a change in the way people use view information on the internet. To improve the ease to use, we developed a tree-style tabbed browsing extension for Google Chrome.

General Terms
Internet

Keywords
tabbed browsing, Chrome extension

1. GOALS AND MOTIVATION
Browsers represent the tool with which a user accesses the information of the internet. As such, it represents the point at which the user's sense of personal organization meets this rich exterior world of information. Moreover, the tools that we use to interact with the world (or, in this case, the virtual world) about us are integrated with our implicit motoric sense of self. This notion, first postulated by the philosopher Martin Heidegger and named "ready-to-hand"[3], has recently been found to extend to our use of electronic tools—in particular, the interfaces with which we use electronic devices [1]. Thus it is reasonable (in fact, it should seem intuitively evident to the reader) to state that the browser represents not only the tool with which we access the internet but the outermost realm of personal organization that we possess in accessing this external repository of information.

Our work was motivated by the vast gap that exists between certain aspects of web design (e.g. the optimization of network design to minimize page load times) and the design of browser interfaces. As the power of computers and the speed of connections increase, users are capable of opening more pages, more quickly, than ever before. Yet the organizational framework that we possess for personally managing these pages has changed little in the past five years. While the remarkably recent introduction of tabbed browsing was certainly an improvement over the previous mode of interacting with the web (in which a user had to open a new window for every web site that he or she wished to visit concurrently), it still provides a very limited interface. This poorly designed interface represents a substantial limiting factor in our ability to efficiently browse the web. A significant amount of human time and mental effort is required to keep track of the context within which individual tabs were opened; this problem is greatly exacerbated when browsing sessions stretch across multiple sittings, or when a browser or computer is restarted and an old session restored.

The guiding philosophy with which we undertook to improve on this situation is that we should design a browser interface that works as much as possible to alleviate the human burden of organizing (or remembering the organization of) individual tabs as they relate to a greater browsing session. Browsing, or the opening of new tabs from old ones, is undertaken as a stream of thought. The user is often led from one topic to the next in a manner that increases in both breadth and depth as the session grows longer; the continual management of this stream of thought is not an intuitive task to humans, but it is a task that can easily be achieved through proper interface design.

Thus our goal in this project was to create an interface for a web browser that achieved this end in an intuitive and simple manner. In particular, we undertook to create an extension for Google's Chrome web browser that provided a visual user interface; we felt that this interface should display the lineage of tabs opened. That is, if a tab is opened from within another tab, then that new tab should be represented as the child of the tab from which it was opened. This should fundamentally free a user from the obligation of remembering the entire stream of consciousness that had led the user to the tab that he or she currently read; instead, a simple glance at the tree created by their browsing would instantaneously reorient the user with respect to the pages that had led him or her to that point.

In addition, such a hierarchical organization could, if customizable, allow users to keep tabs that they found germane to a topic of interest and arrange them in a way that that they felt suitably represented a map of concepts relating to that topic. This would have the benefit of greatly facilitating a user's ability to organize data in a way that makes sense to him or her and that could be extended across mul-

tiple sessions. If such structures could be annotated, saved and shared, users could then assemble resources relating to a given subject and share them with others as a more efficient means of disseminating information that would otherwise have to be reworked into a tutorial or referenced as a collection of links that would either be less organized, less easily organized, or both.

2. TECHNICAL APPROACH

There have been a handful of previous efforts by others to implement similar ideas as browser extensions. However, none has surpassed mere basic improvements to organizational intuitiveness, and worse, most have been abandoned.

For example, "Tab Groups" is a Firefox extension aiming to organize a user's tabs into categories, providing a basic level of structure more relevant to a user's browsing habits. Tab Groups is simply described as "a [F]irefox extension that lets a user organize tabs into groups" [6]. While there is no internal improvement to the structure of tabs within a given group, the organization into groups by itself is still better than the linear, unordered default placement of tabs. The author, in fact, fingered this poor organization as his motivation for the project: default "tab placement is ... less than ideal when the number of tabs" grows large[6]. The project's scope began small ("basic functionality is in place but there [is] a lack of UI niceness") but quickly grew, with its planned features covering covering the simple, such as session management, to the advanced, such as speed optimizations and an overhaul (!). The project remains at version 0.02 and has not been updated since 2007.

"Tab Kit" is another Firefox extension offering many minor improvements, but not much in terms of major organizational structure or more intuitive usability; "Tab Kit makes tabs more efficient for power users, allowing a wide variety of tweaks[.]"[5] It offers an ability to color tabs based on their domains or referers, functionality more limited than that of Tab Groups, and a tree-style organization sorted by history, much like that offered in "Tree Style Tab" (see below). The project is no longer fully compatible with the most recent version of Firefox and has not been updated since 2009.

"Tree Style Tab"[9] is a Firefox extension that simply provides a tree organization of tabs by history. It supports moving child tabs and collapsing subtrees, and it can remember its state across sessions with the help of additional extensions. While not rich in features, at least it has managed to remain up-to-date with regard to versions of Firefox, perhaps because of its limited scope.

Unfortunately, these three Firefox extensions constitute the status of improvements to the intuitiveness of tab organization. (Moreover, note that Google's Chrome browser does not have any counterparts to its name; this is not due to lack of demand[7][10].) The features mitigate somewhat the utter lack of organization by default, but even collectively, they do not approach a complete remedy.

We proposed to create a Chrome extension offering hierarchial or customizable organization, multiple presentation formats, and recommendations based on the current tab. We chose to write an extension for Chrome rather than Firefox because, though Chrome's extension support may be immature and more limiting, we thought that it would be easier for us to work with Chrome when beginning work.

3. DESIGN

3.1 Chrome Extension Structure

Since we decided to build a Chrome Extension[2], we had to follow their structure. Chrome Extensions typically have a background page to save state, a popup page for user interfacing, and an icon to access the extension. In addition, an extension needs a manifest declaring its permissions.

Both the background page and the popup page are HTML pages. The background page remains open the entire time the browser is open and the extension is running, which makes it ideal for saving state. The popup page is created and destroyed as needed.

3.2 Data Structure

We decided to store the data about open tabs in a tree structure, created in JavaScript. This allows us to store the data within the background HTML page.

Each tree object is associated with a parent tree object and a list of children tree objects, as well as a tab object (these may be null). This allows any level of branching that we need, though large trees may result in slow performance.

The current information is associated with a root node in the background page. The popup page can access that root node and from there access all the information in the tree.

We identify unique tabs by a tab id number. This number is provided by Chrome and remains the same for one browser session. In order to update existing tabs, we use a recursive search through the tree to locate the node with that id number.

In addition to the tree, we also store the two most recently selected tabs. Sometimes tabs open in a way that makes it hard to determine from which tab they were opened. This allows us to assume that the most recently selected tab is the one from which the user opened the new tab.

3.3 Event Listeners

In order to get information from the browser, we used the Chrome tabs API[8]. This allowed us to set event listeners, which would be called every time a tab event occured.

When a new tab was created, we added it to our tree. Since sometimes tabs are opened in a way that does not make it easy to determine the tab which opened them (for example, pop-up advertisements that open in new tabs), we assume that the most recently selected tab is the parent tab. Also, if the user opens up an entirely new tab, we assume that he or she is still thinking about the most recent topic and is going to browse on that topic, so we make that tab a child of the most recently selected tab, too.

Sometimes, a user will choose to close a tab on a topic they are no longer interested in. To deal with that situation, we have an event listener for destroyed tags. When a tag is

destroyed, we remove that element from the tree structure and promote all its children to the level it previously had. So, its children become the siblings of its siblings and the children of its parent. Unfortunately, there is currently a bug that causes this process to lose information about the order in which tabs were created, but it still retains the correct ancestry relationships.

As users browse the Internet, the information on each tab changes. Since we display the title of the webpage (or the URL if no title is available), we need to update as information changes. To handle this, we have an event listener that listens for changes in existing tabs. When there is a change in an existing tab, if the title or url changed we update the appropriate entry in our tree. This is especially important, since when we create a new tab, we only know the url since the html hasn't loaded yet. Once the html loads and we know the title, we then automatically update our information.

There is a known bug in the event listeners. If the user does too many actions too quickly, some of the events are dropped. This results in tabs becoming uncoupled from their id numbers, which results in trash nodes in our tree structure. We theorize that this is because we only have one html page with one javascript script, so we only get one thread. Thus, if too many events happen too fast, some of them get dropped.

3.4 User Interface

The user interface is the html popup page. When the user clicks on the icon for the extension, a popup page is created. It is populated from the tree structure stored on the background page.

The tabs are listed in text, with children appearing below parents and indented by two spaces. So, a tabs parent is the lowest tab above it that is two more spaces to the left, a tabs grandparent is the lowest tab above it that is indented four more spaces to the left, etc.

For each tab, we display the tab's title. If no title is known, we use the url. So, as a page loads it will switch from url to title as the html for that page loads.

4. USAGE

4.1 Installing

Since our implemenation is a Chrome Extension[2], the user must already have Chrome installed on his or her machine. To install the Web of Tabs extension, the user needs to first open a Chrome window (see Figure 1).

Then, in order to add an extension to his or her Chrome enviroment, the user must open the Extension tab . This is found via the "Customize and Control" drop-down menu (see Figure 2), accessed by clicking on the wrench icon in the upper right of the Chrome window.

The Extensions Tab is now open (see Figure 3). From this tab, the user can install new extensions, pack extensions, get extension updates, reload extensions, disable extensions, uninstall extensions, and debug extensions. We will only concern ourselves with installing extensions.

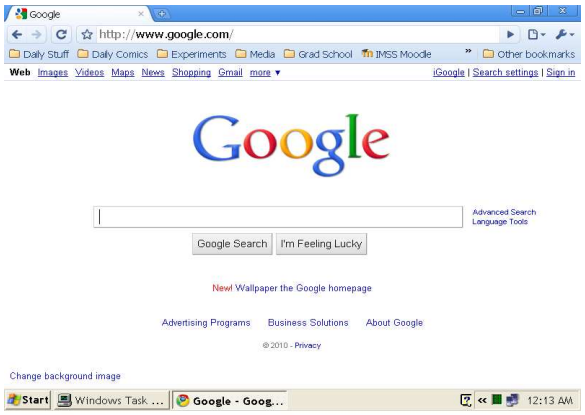

Figure 1: A newly opened window of Google Chrome, an Internet Browser.

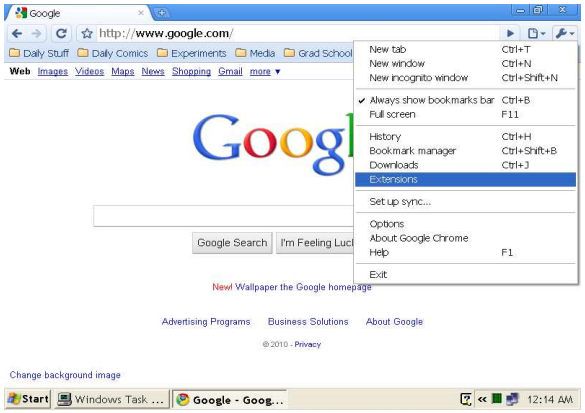

Figure 2: Opening the Extensions tab from the tools drop-down menu.

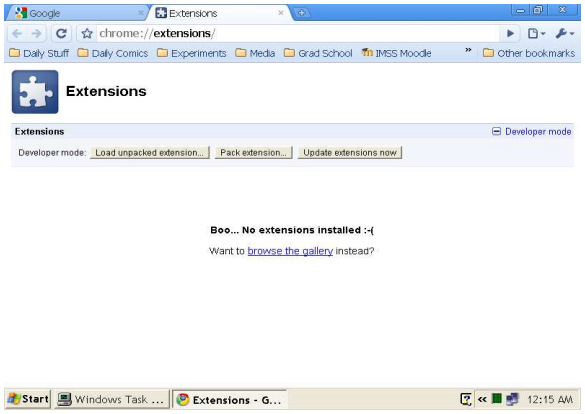

Figure 3: The Extensions Tab. Currently, we have no extensions installed.

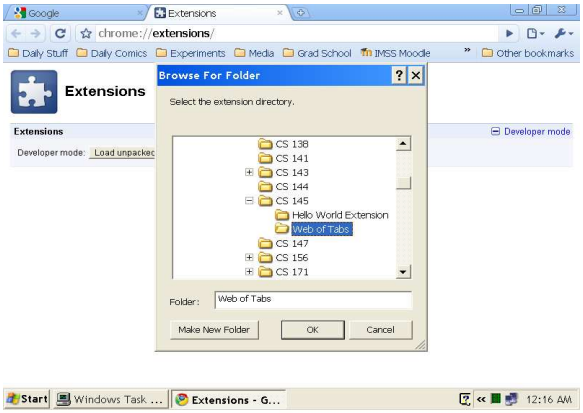

Figure 4: We can load our unpacked Web of Tabs extension from its folder in the file system.

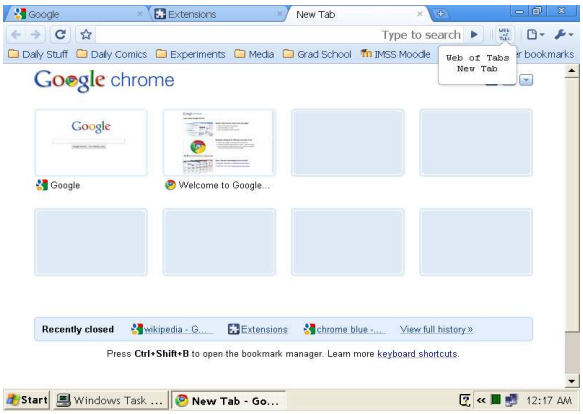

Figure 5: The Web of Tabs extension is now working. A small icon appears in the upper right of the Chrome window, labeled "Web of Tabs". Clicking that brings down a tree of the currently open tabs.

The user can click on the "Load Unpacked Extension" button to load the Web of Tabs extension. Then, the user must navigate to the appropriate folder in the file system (see Figure 4) and select "Ok." From there, the extension will load automatically.

Once the extension has been loaded, a small icon labeled "Web of Tabs" appears in the upper right of the Chrome window (see Figure 5). Clicking on this icon brings up the Web of Tabs popup, which shows the open tabs in a tree-style format. Tabs that were already open when the extension started running (such as the Extensions tab from Figure 3, by necessity) do not appear; only tabs that were created after the extension started running will appear.

The extension will now update its internal data to reflect changes made in the browser. The user is free to surf the Internet, with the added ability to peruse to his or her tabs in a tree structure.

4.2 Browsing

Figure 6: Wikipeida as an example page. As you can see, it appears in the Web of Tabs popup tree structure.

Figure 7: Finland as an example page. The link to Russia is indicated.

As the user goes about his or her browsing, the tree structure of the Web of Tabs will stay updated with the current open tabs. This allows the user to see which tabs were opened from which, creating an ancestry of tabs. A good place to observe this is on Wikipedia (see Figure 6), where each web page has many links to other, relevant web pages.

We can then begin browsing. First, we start by navigating to the page on Finland, which updates the information in the tree (see Figure 7).

Then, we can open the page on Russia as a new tab from the page on Russia (see Figure 8). Since we opened the page on Russia from the page on Finland, the tab containing the Russia web page is the parent of the tab containing the Finland web page. This is indicated to the user by the indentation: the Russia web page is below the Finland webpage and indented further.

After opening many tabs from one starting tab (in this case, all the webpages of countries boardering Russia, see Figure 9), the starting tab has many children. They are all

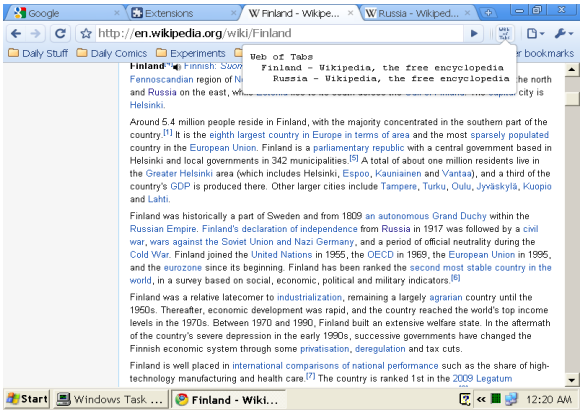

Figure 8: After navigating from the Wikipedia main page to the Finland page and then opening the Russia page in a new tab, the Web of Tabs popup tree structure correctly displays the Russia tab as a child of the Finland tab.

listed below the parent tab in the order in which they were opened. This allows the user to see the titles of all of the tabs, which is not possible at the top of the Chrome window due to space contraints.

If the user opens many tabs, each from the next, he or she can get quite far from the original topic. It is useful for the user, then, to be able to see what path led to the current page. For example, (see Figure 10) casually browsing Wikipedia can lead a user from Finland to Caltech. Being able to trace back one's thought process is very helpful in this kind of situation.

After a brief period of simulated normal browsing, we have some eleven tabs open (see Figure 11). The Web of Tabs allows the user to easily see which tabs are open and which tabs were opened from which. In this example, we the Brown Bear page from the Russia tab, the Cave Bear page from the Brown Bear tab, the Cave Hyena page from the Cave Bear tab, and the Wolly Rhinosaurus page from the Cave Hyena tab, as well as the Exploration, Warrior, Merchant, and Piracy pages from the Viking tab, which was opened from the Russia tab.

The children tabs are kept in the order in which they were opened. In this example (see Figure 12), the tabs 1, 2, 3, and 4 were opened, with 1 first, then 2 second, then 3 third, and 4 last. This is reflected in the order in which they are displayed in the tree.

Users will often close tabs which are no longer useful when other, more useful tabs are still open. If the closing tab has children, they are promoted to the level of the closed tab. In other words, they become siblings of their parent tab's siblings and children of their parent tab's parent (see Figure 13).

When this promotion happens, the newly promoted children are given the same position in the order of siblings that their parent previously occupied. This is to try to preserve infor-

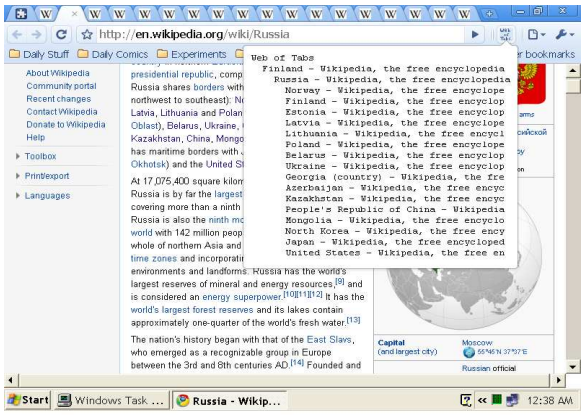

Figure 9: We have opened all of the pages of countries boardering Russia as new tabs. In other words, we have done a partial breadth-first traversal of Wikipedia. All the new tabs are displayed as children of the Russia tab.

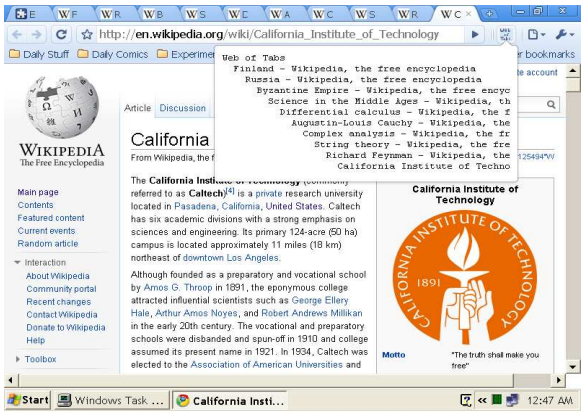

Figure 10: We have opened a series of pages, each from the previously opened page. In other words, we have done a partial depth-first traversal of Wikipedia. Each of the tabs is displayed as the child of the tab from which it was opened.

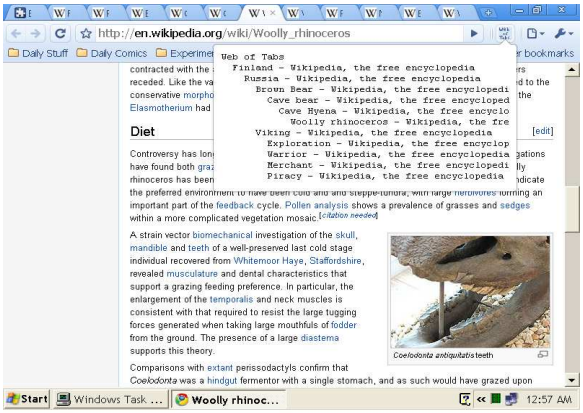

Figure 11: We have opened many tabs in the course of normal browsing, and can easily see which ones were opened from which.

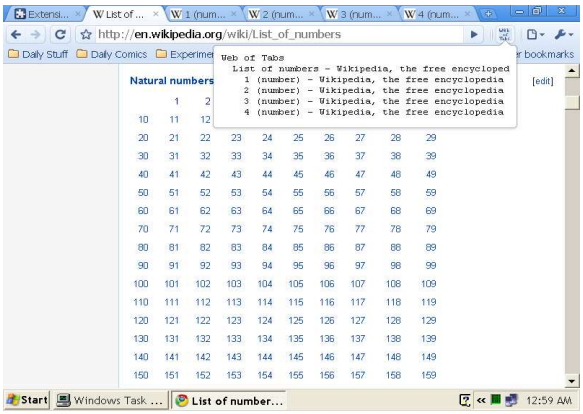

Figure 12: The extension displays the children tabs in the order they were created. In this case, the pages for 1, 2, 3, and 4 were opened in that order, so they are displayed in that same order.

Figure 13: If a tab is closed which has children, all its children are then promoted to the level of the closed tab.

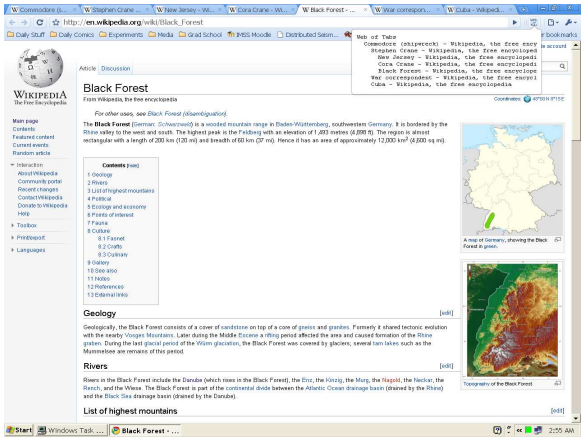

Figure 14: Before a tab is closed and its children are promoted.

Figure 15: After a tab is closed and its children are promoted. Notice that the children are promoted to the same space in the sibling order that the closed tab formerly occupied. Due to a bug, the order of the promoted children is reversed.

mation about the order in which topics were viewed by the user. For example, in figures 14 and 15, the branch starting with Stephen Crane was opened before the branch starting with War Correspondent. So, presumably, the information in the Stephen Crane brach came before the information in the War Correspondent branch, from the user's perspective. It makes sense, then, to keep the information in the Stephen Crane branch before that of the War Correspondent brach when the tree is reorganized. Unfortunately, due to a bug, the order of the newly promoted children is reversed.

As we browse from page to page, the tree keeps the information for each tab updated with its current page's title (see Figure 16). This can cause some confusion, as a parent tab can end up having little to nothing to do with its children tabs, but it is much more useful information that outdated information about pages that are no longer open.

5. FUTURE DEVELOPMENT

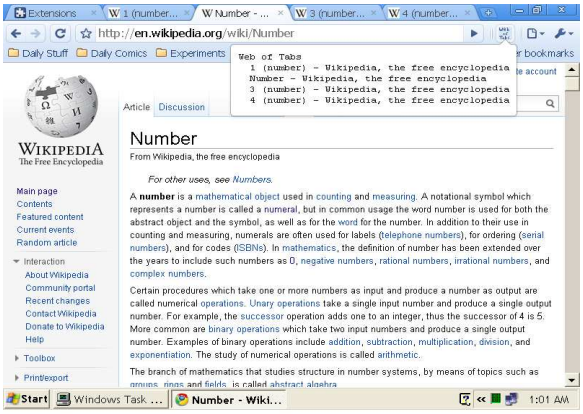

Figure 16: When we browse from website to website within a tab, the tree stays updated. Here, we navigated from the "2" page to the page on numbers.

Development of this project will continue into the summer. As we move forwards, there are an increasing number of possible directions in which we may wish to go; while some conflicting approaches could be reconciled by allowing them to be alternate schemas selectable through options, customization is something of a double-edged sword. Raskin observed that adding customizability inherently makes a user interface more complicated and more difficult to learn in some concrete manner [4]. However, there are of course some options that are simply time-saving conveniences for some users (such as the ability to select a directory for local storage), or that otherwise cater to the tastes of a significant but not overwhelming portion of users. Thus there may be times when we find that options are indeed necessary or beneficial, but we must be prudent so that these options do not in any way convolute the user experience.

The use of multiple windows is a concept that we have not yet attempted to integrate with this project. With our initial goals, it seemed sensible that this application should treat individual windows as entirely separate entities. This may indeed be the proper case for many users; however, there may be others who would prefer to view individual windows as distinct trees that can all be accessed in the same view from the Web of Tabs. As there is likely to be some variation among different users' approaches to distinct windows and to user preferences regarding how those windows should be treated by the application, it would likely be ideal to eventually allow either approach through the customization of user options.

One primary goal is to introduce a simple method for saving and loading browsing sessions and all of the Web of Tabs' information for said sessions. Tradition bookmarks do not allow the addition of third-party information to the data in the bookmark; thus, that avenue seems to be of little value to us now. A second possibility that was investigated, which enables saving and restoring session data locally, would be to store all necessary information in cookies which could then be saved to some specified folder on the user's machine. The major drawback that is presented by this possibility is that cookies are limited to 4 kilobytes in size; thus, one may not have sufficient space to hold all the details that we would need to preserve if the session that is being saved was a reasonably large one.

A third, more promising possibility that we are currently investigating is the use of HTML5's web storage capabilities. This interface would allow us to easily define properties to index; moreover, it allows the storage of megabytes of data on the client side. Although it also enables the user to save data from multiple windows at once, our project's initial aim is such that it would likely be ideal to save the data from each browser window individually. As discussed above, however, it may be preferable to eventually allow either approach and enable users to set their preferences.

The ability to store Web of Tabs data for a browsing session would be useful for a multitude of reasons. The user would be capable of continuing a browsing session without losing the context of the old session (this is an immense problem when tabs are reloaded but there is no organizational structure to help the user to recall the order or significance that he or she ascribed to each page individually). In addition, users would be able to create and share hierarchically organized trees of tabs with one another; this could greatly facilitate the sharing of information that is not contained on one web page but is instead spread across many pages. The advantages of sharing trees of tabs are expanded upon below.

A further goal that we believe to be as conceptually important as the aforementioned ability to save and restore tree structures from previous (or other users') sessions is the ability to interact with and manipulate the tree structure of the Web of Tabs. Fundamentally, this would entail changing the representation of a tab from plain text to an object that contains the tab's title (and, if available, its logo). Double-clicking on this object would bring the user to the page that it represents; clicking and dragging the mouse would enable the user to change the tab's position in the greater tree structure. The former action would require a simple event listener; we already have backend functions in place that are capable of implementing tree customization, but the event listeners (for clicking, dragging, handling relative mouse position, etc.) will likely prove substantially more complicated.

There are a variety of other less significant but still noteworthy features to include with the Web of Tabs applications. Among them are the ability to determine whether the interface appears to the user as a popup or as a new tab (this is comparatively trivial to implement, but we focused on more immediate concerns this term),

As a further direction of future development, it may prove worthwhile to investigate different visual layouts for the tree-like structure of our tab objects. The current representation, in which indentation is used to differentiate between generations, achieves its purpose; however, one of the initial goals of this undertaking was to use a visual layout to make an organizational structure as intuitively apparent as possible to the user. This end is integral to the more abstract objective of using the capabilities of the machine to remove as much as possible any burden of organization from the user. In addition, some alternative representations of the tabs' hi-

erarchy would enable the implementation of other features, such as the use of color-coded lines to conceptually link tabs that are not directly joined by ascendency. This would also enable the tree-style hierarchy to more closely approach a general graph structure.

One final possibility, discussed early in the project's development, would be the extension of this application to suggest recommended tabs from the tab that is currently selected. This would entail a truncated list (perhaps no more than three or four) of web sites, likely the results of a Google search, that we would try to match as closely as possible not only to the topic of the current tab but to the topics of the greater tree of tabs. This would be an intensely interesting problem, but one that may be difficult to solve: inherently, we would seek to give extensions (rather than just repetitions of the current tab's information) that would fit in with the tree as a whole. Whether this could be achieved by some amount of intelligent data-collection from the other available tabs, or whether other methods (such as the mining of data from publically available Web of Tabs files on the internet) would be necessary, is an open question.

6. ACKNOWLEDGMENTS

Thanks to Dr. Steven Low and Minghong Lin for mentoring us on this project.

Many, many thanks to Ryan Witt for suggesting some excellent Javascript tutorials.

7. REFERENCES

[1] D. Dotov, L. Nie, A. Chemero, and V. Brezina. A Demonstration of the Transition from Ready-to-Hand to Unready-to-Hand. *PLoS ONE*, 5(3):e9433, 2010.

[2] *Google Chrome Extensions*, "`http://code.google.com/chrome/extensions/`"

[3] M. Heidegger. *Being and time*. Wiley-Blackwell Hoboken, NJ, USA, 1978.

[4] J. Raskin. *The humane interface: new directions for designing interactive systems*. ACM Press/Addison-Wesley Publishing Co. New York, NY, USA, 2000.

[5] *TabKit*, "`http://code.google.com/p/tabkit/wiki/About`"

[6] *Tab Groups: Firefox Tab Grouping Extension*, "`http://paranoid-androids.com/tabgroups/`"

[7] *Tab Groups and Tab Grouping Please?*, "`http://www.google.com/support/forum/p/Chrome/thread?tid=5a8d6002d0e3520e&hl=en`"

[8] *Tabs - Google Chrome Extensions*, "`http://code.google.com/chrome/extensions/tabs.html`"

[9] *Tree Style Tab*, "`http://piro.sakura.ne.jp/xul/_treestyletab.html.en`"

[10] *Tree-Style Tabbing*, "`http://www.google.com/support/forum/p/Chrome/thread?tid=46072ea13a7726d4&hl=en`"