DTorial: An Interactive Tutorial Framework for Blind Users in a Web 2.0 World

Joshua Hailpern[1], Loretta Guarino Reid[2], Richard Boardman[2]

[1] University of Illinois at Urbana Champaign. 201 N Goodwin Ave, Urbana, IL 61801
[2] Google. 1600 Amphitheatre Pky, Mtn. View, CA 94043, USA
[1] jhailpe2@cs.uiuc.edu, [2][lorettaguarino, richb]@google.com

Abstract. Effective tutorial systems can help promote products by reducing barriers of learning new applications. With dynamic web applications becoming as complex as desktop programs, there is a growing need for online tutorial/help systems. For visually impaired users the key limitations of traditional help systems are 1) poor access to help content with assistive technology, and 2) frequent reliance on videos/images to identify parts of web applications and demonstrate functionality. In this paper, we present a new interaction model, targeted towards screen-reader users, that describes how to embed an interactive tutorial within a web application. The interaction model is demonstrated within a system called DTorial, a fully functional dynamic audio-based tutorial with embedded content. While remaining within the web application, users can rapidly access any tutorial content, injected inline near relevant application controls, allowing them to quickly apply what they just heard to the application itself, without ever losing their position or having to shift windows. The model and implementation are grounded in sighted user help-systems literature and an analysis of screen-reader and Web-Application interactions. Lessons learned from the incremental design and evaluations indicate that providing visually impaired users with dynamic, embedded, interactive audio-based tutorial systems can reduce the barriers to new Web-Applications.

Keywords: Tutorial, Help Systems, Web 2.0, Screen Reader, Blind, Visually Impaired, Interactive Tutorial, Dynamic Content.

1 Introduction

There are many challenges that arise when ensuring equal opportunity access [6, 18, 25] for visually impaired and blind users. Even with state of the art tools (e.g., screen readers), one major hurdle for this community is the adoption of new software. Visually impaired users must rely upon recall to remember the available interface options. Creating such a mental model is a time consuming process. This process is complicated by industry's adoption of Web 2.0 applications (e.g. Dynamic Webmail, Web Document Editing, etc), complex and dynamic online programs that challenge users to use a computer in a completely new way: introducing multiple modes of interaction, including one not in the vernacular of most screen-reader users. While literature would encourage the use of tutorials [23] existing mechanisms (e.g., video,

flash and text+image) rely on visual presentations of context inaccessible to the VIU. Further, traditional Web-based tutorial systems provide poor access to content with assistive technology.

This paper addresses the unique challenges for visually impaired users using web-based tutorial and help systems. We have designed an interaction model specifically addressing the needs of audio-based tutorials systems for Web applications and screen readers. Our interaction model is demonstrated through our fully functional tutorial system, DTorial (**D**ynamic Tu**torial)**. This model and implementation provide a mechanism allowing the VIU to learn the interface of a Web application through embedded tutorial information and an interactive control mechanism. The design is based on a set of interviews with visually-impaired computer users, best practices in existing literature, and an analysis of screen reader interaction with Web 2.0 applications. Our system was further informed through a rapid design and evaluation cycle with 17 VIUs. The foremost contribution of our model is the demonstration of a functional and accessible tutorial that can be integrated easily into the Web-application environment.

2 Definitions and Terminology

Visually impaired individuals range from those who are mildly near or far sighted, to those who are legally blind, to those who have no vision at all. Those with the most vision loss cannot rely upon sight at all to interact with the world around them. Approximately 1.5 million visually-impaired individuals live in the USA [3] and the worldwide statistics put the number at 161 million (about 2.6% of the world population [33]). For the purpose of this study, we define individuals who cannot rely upon sight for computer interaction as being ***V**isually **I**mpaired **U**sers* or VIUs.

Visually impaired individuals have learned to adapt to and augment many aspects of the world around them, for instance, with the Braille system [2]. As computer systems have become ubiquitous, technology also has adapted to meet the needs of these users [25]. One form of assistive technology, the Screen Reader, converts digital text to audio speech or Braille output. Common screen readers include Freedom Scientifics' JAWS, GW Micro's WindowEyes, and Apple's VoiceOver. This software allows users to navigate computer applications and web pages through a series of keyboard commands, while having the content read back to them. This content can be presented as either speech or on a refreshable Braille display [4]. For this experiment, we used audio feedback. We use the term "read" or "reading" to indicate the user is listening to a screen reader converting text to spoken audio. In this experiment we used the JAWS Screen Reader, the most popular screen reader [13].

3 Existing Technology & Limitations

Research in screen readers is ongoing in HCI. Jim Thatcher captured the philosophy best in 1994: "blind users must have access to the same computing environment as their sighted colleagues" [30]. Since then, much work has helped make computers

accessible, with emphasis on improving Web accessibility [23, 26] for screen-reader users. There has also been rapid development of tools to help Web designers check for accessibility [14, 29]. Tools have also been developed to assist in the creation of more accessible Web pages [5, 10]. While new screen-reader based solutions may be invented, such solutions must be disseminated and adopted by the entire community to be effective. However, if new technologies are created and implemented server-side by Web developers, the burden of adopting new assistive technology will not be placed on the millions of existing screen-reader users.

While prior research into help systems and tutorials is rich and provides a strong foundation for future work, existing solutions do not take into account the unique needs and challenges of visually-impaired individuals. Research into tutorials and help systems [7, 19, 21, 22] has explored multimodal interfaces [16], behavior modeling [32], and intelligent help systems [1, 12, 27]. Many solutions have been created utilizing hypertext to provide easy access to help [11, 20]. Though some work has been done on web based help systems [8, 28], it has largely been based in the visual domain (images and Macromedia Flash [24]). Existing tutorials and help systems use video, PDF, Flash, and HTML, while relying heavily on pictures and animation. For VIUs, these modalities are inaccessible. The existing literature has not suitably addressed methods for embedded tutorials in Web 2.0 applications or targeted tutorial techniques for blind users.

The standard form of instruction for VIUs is a separate HTML page containing a tutorial (and discovering the location of said tutorial is not always easy). Most HTML tutorials are not screen-reader friendly, lacking appropriate HTML headings for easy navigation and poor description of how to access/find content with a screen reader. In addition, these tutorials lack context for the content and require frequent switching between windows, often causing users to loose their “place” in both the tutorial and application as they switch between the two windows.

With the advent of dynamic-asynchronous loading in 1999 [31], methods for users to interact with Web pages drastically changed from the world of static HTML. Dynamic Web pages, commonly referred to as Web 2.0 applications [15], often use AJAX to enable content to dynamically change without a page reloading. As a result, these Web applications function like a standard desktop application: the browser acts as a platform on which these new applications run. This facilitates dynamic content without page reloads and additional functionality such as application-based hot keys. Consider a Web e-mail application where a user uses the “c” key to compose an email or the “j” and “k” keys to cycle through the list of emails. Without JavaScript, this functionality would only be available via mouse interactions and not via hot-keys.

Though AJAX provides useful features, Web 2.0 applications add complexity to screen–reader users. Consider functionality in Appendix A, illustrating a JAWS screen-reader interacting with Web pages. A typical screen-reader user operates in two modes: *Forms Mode* (used for text input) and *Virtual Cursor Mode (VCM)* (for page navigation and reading). Interaction is conceptually segregated into these two modalities, making a natural division so users know when to switch modes.

Web 2.0 functionality (check marks in Appendix A) forces users to shift their interaction with the Web application to a third mode, *PC Cursor Mode* (*PCM)*, to use Web-application specific keyboard commands. *PCM* stops JAWS from capturing keystrokes and allows the user to invoke application-specific hot keys. However,

PCM is not familiar to the typical screen-reader user. This shift in the way the screen reader is operated is needed so that Web applications can capture user keyboard input (normally intercepted by JAWS for screen-reader functionality). Unfortunately *PCM*, also lacks audio feedback for users (see Appendix A).

For VIUs, Web 2.0 applications not only pose the common challenge of learning a new application, but also a burden in requiring new screen-reader interactions. VIUs must overcome both hurdles simultaneously. Thus, a Web-based tutorial system, designed for VIUs' needs, is critical for accessibility of ubiquitous Web Applications.

4 Interviews with Visually Impaired Users

There are many guides for creating tutorials on the desktop [7, 19, 21, 22] and for the Web [8, 28], but less is known about how VIUs approach computer usage and their concerns with adopting new software. To gain further insight, we conducted five 2-hour interviews with members of the visually impaired community. Participants were all screen-reader users from Silicon Valley (remunerated $75/hour). We attempted to recruit a representative user pool, individuals whose primary profession was not high technology. Each interview focused on participant's computer usage specifically, file organization, application management, and inter-personal communication.

Interviews were transcribed and video taped. All five participants reported large effort required to adopt new applications. However, each participant talked about new software adoption from a different perspective. One spoke about waiting to take a class to learn new software, while another said that she needed to wait for a large amount of time to be available to learn a new piece of software due to trial and error. Another individual wished to see a "guardian angel" that would show him around his programs and answer his questions. Participant issues revolved around the screen reader world-view that comes from memorization of key commands and Web page/application layout.

From our observations, we concluded that an accessible Web-based tutorial and help system would improve Web application adoption (akin to the findings of [23]). An examination of the existing practices of sighted users, detailed analysis of AJAX/screen-reader technologies, and comments from interviewees lead us to a proposed set of 4 key requirements that such a system should have:

> •**Embedded Content** - Because layout and context is critical to users, any help system should provide its content in the context of the target application to reduce chances of losing position from window switching.
>
> •**Interactive and Dynamic Content** - Users should be able to quickly gain access to and change their tutorial/help system content while in their application, thus reducing trial-and-error by allowing topic lookup as an easy alternative. *Note: while the concept of dynamic content is not new* [9], *the application to a non-visual domain is novel.*

•**Audio Based Content** - Because users rely entirely on audio cues, developers should ensure that content is presented in such a way to be fully understood through audio only and existing screen readers.

•**Mitigate AJAX** - Due to the complications created by screen reader interaction with Web 2.0 applications, tutorial systems must address this new complexity (for their own interaction model and the model of the target application).

5 DTorial and Incremental Design/Evaluation

In response to the complexity of learning new applications and the elaborate Web 2.0 + screen reader interaction techniques, we developed an initial design for embedded Web-based help systems for VIUs. In order to develop and evaluate our design, we built the DTorial system. By evaluating, testing and incrementally improving DTorial, we are able to analyze user behavior, reaction, and resulting design changes to create a higher-level interaction model.

5.1 DTorial

The DTorial system is a fully functional embedded tutorial, built using JavaScript. We used Grease Monkey to implement DTorial on top of Google's Web 2.0 Gmail, chosen because email is ubiquitous and is crucial to most individuals. Participants in our initial interviews said that email is empowering and central to their communications at work and at home. We used a set of JavaScript libraries called AxsJAX [10] to facilitate audio feedback and audio alerts.

DTorial works by moving a user around a Web page while simultaneously providing the user with audio feedback and describing the new content, what to expect, and how to interact. In other words, DTorial teaches the user how to interact with the computer while using the tool they are learning. As a result, the user is easily able to apply the skills learned without switching environments.

For example, consider a fictional VIU, Alice. When Alice wishes to understand the concept of the *Folders List* in Gmail (Inbox, Drafts, etc), she locates the *Drop Down List* on Gmail's webpage entitled "Tutorials." Alice selects the tutorial entitled "Folders" and then presses the *Go Button* to launch her content. The web browser's focus is automatically moved, via JavaScript, from her current element (e.g. *Go Button*) to the *Folder List* in Gmail. Simultaneously, tutorial text explaining *Folders* is injected into the DOM, at the top of the *Folder list*, within the now focused region. While having no visual impact to the layout of the page (using CSS markup), screen reader users like Alice can now access the entire body of injected text, including navigation by headings, lists, and other HTML elements. Alice can immediately explore the tutorial and test out her new knowledge without having to leave the instructional guide, because the guide AND environment, are side-by-side. DTorial further blurs the line between instruction and execution by ensuring a mechanism for re-reading the instructions at any time. It should be noted that the method for injecting

text and re-reading content evolved through the incremental design process, and this example is representative of the final iteration of the DTorial system.

5.2 Incremental Design and Evaluation Methodology

In order to test learning via DTorial, we compared it against the standard form of instruction for VIUs, separate HTML web pages containing a tutorial. While most HTML online tutorials are not screen-reader friendly, we eliminated bias towards DTorial by ensured that the HTML tutorial had marked headers, no references to video or images, and provided the same content as the interactive counterpart.

We followed a rapid cycle of evaluation and redesign. Visually impaired subjects were recruited from centers and organizations for the visually impaired in California's Silicon Valley and San Francisco. See Table 1 for demographics. Though we recruited 20 users, 17 individuals participated (eight men and nine women). Two were determined to be ineligible for the study when it was discovered that they were not visually impaired. One was removed from the study due to technical difficulties that arose during the session. A typical session with one subject lasted for one-and-a-half hours. Subjects were remunerated ($75/hour). Participants had no prior experience with Gmail, though some had accounts that were forwarded to desktop email clients.

Tutorial text was based on Gmail's Getting Started guide. We limited Gmail's feature set to Compose Mail, Inbox, Drafts, Spam, Trash, and Message Threads so we could focus on the tutorial and the learning experience. Because Web 2.0 applications require users to be in *PCM* when using hot-keys, we added audio feedback via AxsJAX, to increase accessibility (e.g., so that audio-based alerts were spoken when pages changed and updated).

During a session, a participant was exposed to each tutorial for approximately 30 minutes. The participant was instructed to "do as you normally would, as if trying this out for the first time on your own and as if we were not here." During a session with the HTML tutorial, a participant was provided with two windows, one pointing to Gmail and the other to the tutorial. After an exposure, each participant was asked a series of questions focusing on usability of the tutorial, accessibility of the tutorial content, how much the participant had learned about using the application, and methods for improvement. Following exposure to both tutorials, a series of questions were asked comparing the two methods. Participants were asked to indicate and justify a preference between the two tutorials.

The order of exposure was randomized to overcome learning effects. Overall, eight

Table 1. Demographic in Iterative Usability Study

Number of Users	17		
Mean Age	40 years		
Number of Users Per Age Group	20-29 (4)	30-39 (2)	40-49 (4)
	50-59 (3)	60-69 (4)	
Computer Experience (years)	1.5-23 Average 12.1 years		
JAWS Experience (years)	1.5-16 Average of 10.5 years		
Number of Users Exposed to DTorial First	8		
Number of Users Exposed to HTML Tutorial First	9		

participants were shown DTorial as the first exposure, and nine were shown the HTML version as the first exposure. DTorial went through five iterations, and averaged four participants per cycle. Table 1 summarizes user demographics, experience, and conditions. DTorial went through five major design and evaluation cycles. Each cycle had 3-4 users. Revisions after each cycle were based directly upon user feedback and researcher observation.

6. Findings from Design and Evaluation Cycle

From the incremental design and evaluation with 17 real world users, we analyzed the behavior and design changes of DTorial and created a high-level interaction model. This set of design requirements is based on the results from our users and our iterative design and evaluation cycle.

At the conclusion of 17 studies, most participants were extremely positive about the prospect of using DTorial for their day-to-day learning. Through the design cycle, we noticed a distinctive shift in comments from users. Originally, feedback focused on the difficulty in switching modes and the complexity of Web 2.0 applications. At the conclusion of our studies, feedback was primarily about phrasing of the content, what topics to discuss, and adding additional functionality for a more robust user experience. Overall, 12 of our users wanted to use DTorial in their learning process. Out of the 5 who did not, all but one used the earliest versions of our software, without the benefits of our incremental improvements. Moreover, their concerns were addressed in iterative process of designing DTorial benefiting the later participants. One participant who teaches other VIUs said the following about DTorial:

> *This [interactive tutorial] is a very smoothly integrated environment for learning and getting your task accomplish… I think they are both good, but I think the learning curve will be a little bit more [steep] in [the HTML tutorial]. – P7*

The remainder of this section, we will focus on what Web 2.0 designers can do to help incorporate tutorials into their applications.

6.1 User Found 3 Modes of Interaction Confusing

The first requirement is based upon users' aggravation with the complexity of Gmail when using a screen reader: not all functionality was evenly distributed between the three Screen Reader modes. In Appendix A, we notice that users can **always** read in *VCM* and can control the Web application in **both** *VCM* and *PCM*. To control some features of Gmail, a mode switch was required (e.g., using a hot-key), while others times, mode switches were optional (e.g., accessing a check box). One user characterized this complexity by saying:

> *You have [the Virtual Cursor] off to read, and off to perform some of the keyboard commands, and you have other commands that you have to use when the virtual cursor is on, and then you have to*

> *remember that you have to have the virtual cursor on to turn forms mode on, which confused me a couple of times... it just seems to be a lot of steps.* - PP1

Because users relied on memory to recall hot keys and link locations, they were unable to remember every mechanism for content access, relying on a smattering of recalled access methods across the different modes.

> **RECOMMENDATION:**
> **Enable Interaction Based Modes (Reading & Control Mode)**

Due to the interaction limitations imposed upon screen-reader users when using Web 2.0 applications, user interaction must be simplified to easily categorized modes of interaction, to reduce user recall burden. To this end, we made a sharp division between control commands and reading commands to eliminate the mixing of metaphors (Appendix B). During the tutorial, we dubbed *PCM* as **Control Mode** and *VCM* as **Reading Mode** [17]. Navigation of Gmail was relegated to hot keys accessed in Control Mode. Though there may be multiple ways to achieve a goal (both with hot keys and by clicking links), tutorials (and applications) should clearly differentiate between reading (both tutorial and emails) and interacting with the application.

While audio feedback was limited in **Control Mode**, AxsJAX was used to provide audio feedback. Appendix B illustrates this modal breakdown. We ensured that all information and controls presented to users fit in one of two discrete modes. As the user becomes advanced, lines between the modes can be blurred. However, for novice Web 2.0 users, ensuring a clear distinction is essential. It will make the difference between the application seeming Accessible and Inaccessible.

6.2 Users wanted non linear access to content

Providing an easy mechanism for the user to navigate content by topic before reading is essential. If the user is presented with a long list of topics and their content (as in HTML tutorials), he can be overwhelmed and inclined to skim through rather than focus on learning specific details. One user summarized why she liked DTorial better:

> *If you got [the HTML tutorial] in one window and you are reading, you tend to zone out and read-read-read, and then you go back to the other window to try it out and say 'now what did I just read?'... I just read it and I forgot it!* - PP12

Initially, DTorial had users moved from topic to topic via hotkey (move forward, back, and repeat), forcing them to iterate over all topics in turn. However, this resulted in the same difficulty as iterating over the long HTML tutorial.

> **RECOMMENDATION:**
> **Facilitate Random Access to Tutorial Content via Quick Access Technique (e.g. Search, Drop Down List) within Application**

In a later iteration, DTorial avoided the nonlinear content access difficulty through a combo-box mechanism. Users could quickly iterate over possible tutorial topics, and from that list, select the instructional segment they wish to explore. This resulted

in one of the most successful aspect of DTorial, the ability to quickly select and learn one specific aspect of a program via tutorial. The random content access system can be further enhanced through tutorial searching. Designers must remember that "finding" is universal. Blind or sighted, users want to find an answer to a specific question. Allowing random access to content ensure a quick resolution.

6.3 Users Felt that They Did Not Know What was Occurring and were Not in Control of Navigation

In an AJAX application where pages change dynamically without page reloads, the screen-reader user relies upon the application to notify him when content changes. Though this is critical for Web 2.0 application design, it also influences the design of tutorials. For example, early in the DTorial design cycles, all tutorial content was injected directly to the screen reader. However, when text was injected in this manner, users were unable to reread in their traditional manner; reading by paragraph, sentence, or even words. One user stated that:

> *I didn't have to read a whole section at a time, I could go back through an read word by word, line by line... - PP1*

Lack of notifications when content changed, and the inability to read at one's own pace, caused considerable distress and made users feel like the "were not in control."

> **RECOMMENDATION:**
> **Always Keep Users Informed and in Control With Feedback**

Tutorials and Web 2.0 applications can address this problem though a solution like AxsJAX and inject content to be read when dynamic changes are made to the DOM. Further, when the screen-reader user wants to jump to a chapter, he must be able to select the new topic, and be notified that he is at a new location on the page. In addition, content in the tutorial must be book-ended, with both a header up front and a textual warning at the end that he is leaving the tutorial and returning to the Web application. This allows him to easily find the content again, and alerts him when he is leaving the tutorial information and reentering the application.

In the above example relating to DTorial content, we modified the system to inject tutorial text directly into the DOM, so that browsers treated it as actual page text, permitting users to read the tutorial content in the same manner as they read other web pages (whole text, line-by-line, and/or word-by-word). The text was inserted directly above the topic of discussion (e.g., the chapter on folders would be injected just before the folder list). As users moved from chapter to chapter, the old tutorial text would be removed and new text would be added. Yet as long as the user was in one section/chapter/topic of the tutorial, the content remained for reference at any point, so users could try out what they learned along the way. We further demonstrated that the text could be hidden with CSS and still be accessed by screen readers, thus making the tutorial only visible to a screen reader user.

6.4 Users Requested Both Upfront Tutorial and Embedded Help System

At the conclusion of our user studies many participants liked DTorial. However most requested an up-front tutorial document in HTML to give an overview of many or most of the topics. Getting a good all-in-one-read tour is necessary before exploring. The following quote illustrates why presenting a mental map of page content upfront is so important to users:

> *One key component of learning for a visually impaired person is orientation; I like to know where I am in the big picture* - PP4

However, an HTML tutorial alone still presents many of the difficulties experienced with the current state of online help systems.

> **RECOMMENDATION:**
> **Provide Tutorials with Embedded Quick Help**

Users explicitly requested a system design with both an upfront HTML tutorial (as a getting started guide) with embedded quick help system (that is accessable within the web application via search, list, or dropdown menu). Having a quick in-line reference like DTorial is a complement to facilitate trial and error, experimentation, quickly learning how to execute new functions, or get refreshers on forgotten features without leaving the application. One individual described having both systems as:

> *It would be the Cadillac, you would give me both of these... give it all to me and charge me not extra for it. – P4*

Another user described how having DTorial as a persistent element of the interface can provide constant support:

> *It seems like the fact that you have gotten rid of the complexity of having to change windows to do anything. I think the other thing is that this type of setup you could leave around, even if you, as you got more familiar with it, since it has the hide tutorial option, and if you did forget something... So basically, it seems that if you did temporary forget something, or want to go back look up a function that you didn't quite remember or didn't use often. The fact that you had that there would be really helpful – P17*

6.5 Users Complain about Readability

Since VIUs cannot skim a page visually, screen reader users are dependent on features in a screen reader to help them navigate the page to the textual content. Users must either listen to all the content in order, or use methods to stop and skip around on the page. One user specifically stated that:

> *The less irrelevant information the better… The less information, as long as you're deleting irrelevant information, I think it is better [or a JAWS user]*

Because listening is slower than reading, verbose language slows reading down, and if rushed, may skip sections not realizing where the vital material is located.

RECOMMENDATION:
Facilitate Methods of Reading Content through Screen Readers

Though often overlooked, following well-accepted accessibility practices when creating content is critical (e.g., bulleted lists that are clear, concise, terse, and to the point). For the visually impaired community, markers (such as headings and bullets) not only serve as demarcation between content and sections, but also facilitate aural skimming of content. Therefore, textual content should be well marked and language should be task based, in step-by-step format, simple, concise, and to the point to facilitate screen reader scanning.

7. Sighted Users and Future Work

We propose two main thrusts for future research to continue to explore this metaphor of the interactive tutorial. First, a system like DTorial should be tested in a full-scale system and deployed to numerous individuals. Testing the scalability and usability over a longer term will help validate this design in day-to-day scenarios.

Following the completion of the design cycle, we speculated on how a similar model on interactive tutorial could be applied to the sighted community. This form of DTorial would function by highlighting and enlarging the selected area, while providing an overlay of tutorial content for the user (Appendix C). This would provide the same functionality for the sighted user as for the screen-reader user, but in the modality that is most applicable. We created both an interactive prototype and a static visual mock-up of such a system. We took our design to the cafeteria of a large Silicon Valley, CA. company and asked both technical and non-technical employees their opinion was of such a model. Though many mentioned that they would like to have search capabilities and a version with all the content in one place (like the user of a screen reader), 100% of the 15 individuals asked felt this would be a "fantastic" solution. "Great idea, love it!" Others specifically mentioned that "[this tutorial] would be great for my parents." We encourage additional work to test the DTorial model for the sighted user and to examine how lessons learned from web accessibility can help all individuals learn to use new and complex web applications.

8. Conclusions

Equal accessibility for all users is a critical part in software design and deployment. With the advent of Web 2.0 technology, visually impaired individuals are forced to use their computer in new ways. One user described this paradigm shift as:

> *Driving on the opposite side in Europe. Because every time they go into this program they are going to have to junk everything that is in their head… - P10*

Because there is such a burden on these users for adopting new technology, it is incumbent upon us as researchers to explore new techniques and technologies to help educate and inform users on how to use new software and the changing Internet.

This paper presents a new interaction model and its demonstration, called DTorial. Unlike traditional tutorials that rely on images and video (inaccessible to VIUs), DTorial is an interactive guide to a Web 2.0 application. DTorial was iterated upon through five rounds of evaluation and redesign, with a total of 17 VIUs. Not only were VIUs excited by our design, but there are interesting applications to the sighted community as well.

Using the framework outlined by DTorial, we believe that Web 2.0 applications can move from inaccessible obstacles to the visually impaired user, to just another application to accomplish their goals. Our design cycle with visually impaired users, indicate the potential effectiveness of DTorial and our model for teaching screen-reader users new web applications.

9. Acknowledgements

We would like to thank all of our participants, Sensory Access Foundation, VISTA Center for the Blind, and Lighthouse for the Blind.

10. References

[1] Aaronson, A. and Carroll, J. M. Intelligent help in a one-shot dialog: a protocol study. In *Proceedings of the ACM SIGCHI Bulletin* (Toronto, Ontario, Canada, 1987). ACM, 1987.

[2] American Foundation for the Blind. Braille. http://www.afb.org/Section.asp?SectionID=6 Accessed 2008

[3] American Foundation for the Blind. Facts and Figures on Americans with Vision Loss. http://www.afb.org/Section.asp?SectionID=15&DocumentID=4398 Accessed

[4] Becker, J. V., Becker, D. A., Hinton, D. E. and Hugh G. Anderson, J.*Braille computer monitor*(6700553). Unites States Patent Office, USA 2004.

[5] Bigham, J. P. and Ladner, R. E. Accessmonkey: a collaborative scripting framework for web users and developers. In *Proceedings of the W4A 2007* (Banff, Canada, 2007). ACM, 2007.

[6] Caldwell, B., Cooper, M., Guarino-Reid, L. and Vanderheiden, G. *Web Content Accessibility Guidelines 2.0*. 2008.

[7] Carroll, J. M. and McKendree, J. Interface design issues for advice-giving expert systems. *Commun. ACM*, 30, 1 1987), 14-32.

[8] Chamberland, L. Componentization of HTML-based online help. In *Proceedings of the International conference on Computer documentation* (New Orleans, Louisiana, United States, 1999). ACM, 1999.

[9] Chang, B., Mackinlay, J., Zellweger, P. and Igarashi, T. A negotiation architecture for fluid documents. In *Proceedings of the UIST 1998* (San Francisco, CA, 1998). ACM New York, NY, USA, 1998.

[10] Chen, C. L. and Raman, T. V. AxsJAX: a talking translation bot using google IM: bringing web-2.0 applications to life. In *Proceedings of the W4A* (Beijing, China, 2008). ACM, 2008.

[11] Comeau, J.-M. and Milton, P. R. A window-based help, tutorial and documentation system. In *Proceedings of SIGDOC* (Waterloo, Ontario, Canada, 1993). ACM, 1993.
[12] Fischer, G., Lemke, A. and Schwab, T. Knowledge-based help systems. In *Proceedings of CHI 1985* (San Francisco, California, United States, 1985). ACM, 1985.
[13] Freedom Scientific. JAWS for Windows ® Screen Reading Software. http://www.freedomscientific.com/products/fs/jaws-product-page.asp Accessed
[14] Fukuda, K., Saito, S., Takagi, H. and Asakawa, C. Proposing new metrics to evaluate web usability for the blind. In *Proceedings of CHI '05* (Portland, OR, 2005). ACM, 2005.
[15] Garrett, J. J. Ajax: A New Approach to Web Applications. adaptive path, February 18, 2005. http://www.adaptivepath.com/ideas/essays/archives/000385.php, 2005.
[16] Greyling, J. H. and Calitz, A. P. The development of a computerised multimedia tutorial system for a diverse student population. In *Proceedings of AFRIGRAPH* (Cape Town, South Africa, 2003). ACM, 2003.
[17] Hailpern, J., Guarino Reid, L., Boardman, R. and Annam, S. *WEB 2.0: Blind to an Accessible New World*. ACM, City, 2009.
[18] Harkin, T. AMERICANS WITH DISABILITIES ACT OF 1990. U. S. Senate, 1990.
[19] Houghton, R. C. Online help systems: a conspectus. *Commun. ACM*, 27, 2 1984), 126-133.
[20] Kantner, L. and Rusinsky, L. Designing a WinHelp project for quick conversion to lowest-common-denominator HTML-based help: a case study. In *Proceedings of SIGDOC* (Quebec,, Canada, 1998). ACM, 1998.
[21] Kearsley, G. *Online help systems: design and implementation.* Ablex Publishing Corp., 1988.
[22] Kearsley, G., Campbell, R. L., Elkerton, J., Judd, W. and Walker, J. Online help systems: design and implementation issues (panel). In *Proceedings of CHI '88* (Washington, D.C., United States, 1988). ACM, 1988.
[23] Landram, F. G. Computer tutorials for introductory statistics. In *Proceedings of CCSC '01* (Branson, Missouri, United States, 2001). Consortium for Computing Sciences in Colleges, 2001.
[24] Lee, K. and Lee, D. H. An online help framework for web applications. In *Proceedings of the International Conference on Design of Communication* (El Paso, Texas, USA, 2007). ACM, 2007.
[25] McKeon, H. P. The Rehabilitation Act Amendments (Section 508). U. S. Congress, 1998.
[26] Miyashita, H., Sato, D., Takagi, H. and Asakawa, C. Making multimedia content accessible for screen reader users. In *Proceedings of the W4A 2007* (Banff, Canada, 2007). ACM, 2007.
[27] Patrick, A. and McGurgan, A. One proven methodology for designing robust online help systems. In *Proceedings of the International conference on Systems documentation* (Waterloo, Ontario, Canada, 1993). ACM, 1993.
[28] Rintjema, L. and Warburton, K. Creating an HTML help system for web-based products. In *Proceedings of SIGDOC* (Quebec, Quebec, Canada, 1998). ACM, 1998.
[29] Takagi, H., Asakawa, C., Fukuda, K. and Maeda, J. Accessibility designer: visualizing usability for the blind. In *Proceedings of the SIGACCESS 2004* (Atlanta, GA, USA, 2004). ACM, 2004.
[30] Thatcher, J. Screen reader/2: access to OS/2 and the graphical user interface. In *Proceedings of the Assets '94* (Marina Del Rey, CA, 1994, 1994). ACM, 1994.
[31] W3C. HTML 4.01 Specification - IFRAME. http://www.w3.org/TR/html4/present/frames.html#h-16.5 Accessed Aug 2008
[32] Willis, M. Building effective help systems: modelling human help seeking behaviour. In *Proceedings of the CHISIG of Australia 2006* (Sydney, Australia, 2006). ACM, 2006.
[33] World Health Organization. WHO | Magnitude and causes of visual impairment. http://www.who.int/mediacentre/factsheets/fs282/en/ Accessed Aug 2008

Appendix

Appendix A:
A breakdown of user functionality associated with providing technology and the Screen Reader Mode. Non-shaded region is tradition web browsing/computer usage. Shaded area is Web Application interaction.

Features	Virtual Cursor Mode	Forms Mode	PC Cursor Mode
Access Check Boxes	●	●	●
Navigate by TAB key	●	●	●
Access to Buttons	●	●	●
Execute Links	●	●	●
Read & Re-read Text *	●	●	
Access Combo Boxes	●	●	
Text Input		●	
JAWS Find	●		
Navigate by Element Type	●		
Navigate by Arrow Keys	●		
Navigate using Headings	●		
Navigate by page links	●		
Easy Tables Navigation	●		
JAWS Goto Line	●		
Read bold, italics, etc.	●		
Web Application Hot Keys **			✓
JavaScript caused Text input			✓
Move Focus via JavaScript			✓
JavaScript changes DOM ***			✓
Utilize JavaScript State			✓
Generate spoken text ****			✘

● JAWS Web Browsing ✓ Web 2.0 Features ✘ AxsJAX Features

* In forms mode, only re-read text in edit areas
** *Virtual Cursor Off* prevents conflict with Screen Reader Keys
*** without audio feedback
**** cannot go word-by-word or line by line

Appendix B:
Screen Reader modality and features associated with Control and Reading Mode metaphor. Shaded area represents features used to access tutorial content.

Features	Reading Mode	Control Mode
Access Check Boxes	●	
Navigate by TAB key	●	
Access to Buttons	●	
Execute Links	●	
Read & Re-read Text *	●	
Access Combo Boxes	●	
Text Input	✓	
JAWS Find	●	
Navigate by Element Type	●	
Navigate by Arrow Keys	●	
Navigate using Headings	●	
Navigate by page links	●	
Easy Tables Navigation	●	
JAWS Goto Line	●	
Read bold, italics, etc.	●	
Web Application Hot Keys **		✘
JavaScript caused Text input		✘
Move Focus via JavaScript		✘
JavaScript changes DOM ***		✘
Utilize JavaScript State		✘
Generate spoken text ****		✘

● Virtual Cursor Mode ✓ Forms Mode ✘ PC Cursor Mode

* In forms mode, only re-read text in edit areas
** *Virtual Cursor Off* prevents conflict with Screen Reader Keys
*** without audio feedback
**** cannot go word-by-word or line by line

Appendix C:
Mockup of DTorial in a visual domain. Notice how the folder list has been highlighted, and expanded in size, while the call out presents the tutorial's textual content.