Enabling End Users to Independently Build Accessibility into the Web

Jeffrey P. Bigham
Computer Science and Engineering
DUB Group
University of Washington
jbigham@cs.washington.edu
http://www.cs.washington.edu/homes/jbigham/

ABSTRACT

Providing an accessible, usable experience to all web users has been a challenge since the inception of the web. Developers of web content target their designs to visual display, and expect input to come from both a mouse and a keyboard. Providing an accessible experience is much more than an all-or-nothing problem, it requires considering a spectrum of problems. This chapter considers the following three levels in the hierarchy of accessibility problems and how end users can contribute to improving them: (i) making access to content possible, (ii) making access to content usable, and (iii) making access available wherever users want it or need it. End user programming is an attractive solution to improving accessibility because it directly connects users with the incentive to improve content with the ability to improve it.

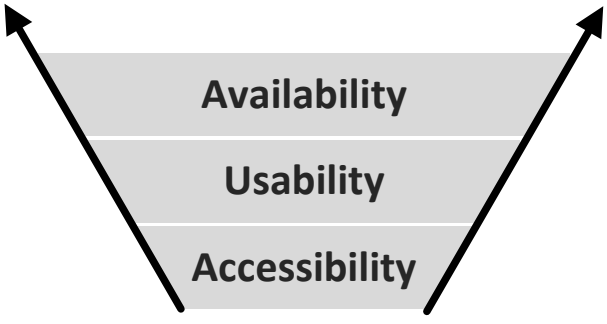

Figure 1: Achieving web accessibility requires more than simply making it possible for users with diverse abilities to access content. Accessibility is the foundation on which the usability and availability of access rests.

INTRODUCTION

End users need to be able to independently program accessibility into the web. Web accessibility concerns have existed for nearly as long as the web has existed. This chapter looks at the impact users can have on their own web experiences by contributing to their accessibility, either through direct improvement or by helping to inform developers of problems. This chapter primarily targets improved accessibility for blind web users, but the examples can be extended to improving access for people with different disabilities.

Early access technology dealt with the text-only content of the early web reasonably well, but started having difficulties as early as the introduction of the image (IMG) tag, which brought multimedia content to the web. Ever since the first drafts of the HTML standard, the alt attribute was provided as a way to provide a description of images, but, nevertheless, as of 2006, less than half of the informative images on popular web sites were assigned alternative text [5]. Requiring web developers to build accessibility into their content has not proven to be a reliable solution to achieving accessibility.

Instead of full reliance on web developers, we envision a web that all users can actively shape to work better for them. For disabled computer users, the web offers the promise of endless content easily converted to an accessible format, but barriers to achieving this full potential remain for anyone accessing the web using a non-standard display, keyboard and mouse. Some web content is encoded visually assuming a certain display size, content can be difficult or inefficient to access with assistive technology, and access almost always depends on the

ability to install special access software, which users often lack the permission to do.

The web is not designed with blind web users in mind, but is instead designed targeting mouse-driven visual displays. Blind web users access their computers and web content using non-visual access software called screen readers, which (i) convert information typically displayed visually to a linear stream of output in either synthesized voice or refreshable Braille, and (ii) provide a large number of shortcut keys designed to help make searching that linear space more efficient.

Access for blind web users remains inefficient, slow, and often frustrating. Accessibility efforts have focused on making access possible, but the resulting interfaces remain unintuitive to use. Tools lack an understanding of the semantics of content, and, therefore, have trouble conveying it in a meaningful way. This chapter overviews our work in collaborative accessibility, which we are exploring to enable blind web users to independently improve the web to better suit their needs and directly address the accessibility problems they experience.

Web developers, who can choose to create more accessible content, are commonly blamed for these problems. As an example, *target.com* recently lost a multi-million dollar lawsuit against the National Federation of the Blind because of its inaccessible online storefront. Although making access possible is relatively straightforward (and Target could have made simple improvements to make its site accessible), web developers have, in general, proven unable to reliably create accessible content.

Part of the problem is that it is difficult to understand what might hinder access someone different than you might access content and predict what problems they might have. The result is that even when developers try to follow accessibility standards in order to make it possible for people with disabilities (or someone using a small-screen device) to use their sites, access is still a frustrating and unintuitive experience. Most developers of content are not disabled themselves, and so many do not know what a disabled user might need or want out of an interface. Accessibility issues involve not only technical considerations, but issues of cost to implement or rework existing content. Just as the best visual designs require keen subjective construction, so do the best accessible designs require substantial design skill.

End users understand when content is difficult for them to access, but they often lack the tools and technical knowledge to improve content to work better for them. To address this shortcoming in existing tools, we have developed socially-driven tools to enable end users to independently build accessibility into the web and to share the improvements that they make with others. Our work has focused on non-visual access both because of the incredible potential for social impact in this space and because we believe this to be an extreme case that can inform improvements for users with different requirements. The challenges addressed are applicable to a wide variety of end user programming tools for improving web content according to an individual's abilities and preferences.

We have divided accessibility problems into the following three categories and associated research questions:

(i) **Accessibility –**

How can users access rich content regardless of ability?

Multimedia content lacking alternatives is not accessible programmatically or easily conveyed non-visually without explicit annotation. Content is often not accessible using only the keyboard, which makes it difficult for keyboard users.

(ii) **Usability –**

How can users help one another more effectively complete tasks on the web?

Complex content can make accomplishing tasks inefficient for blind users and confusing for cognitively-disabled users. Content in a second language can be difficult to understand.

(iii) **Availability –**

How can the access technology and social improvements provided by users be provided to everyone that needs them where and when they need them?

Access technology is not available on most computers, including lock-down public terminals. Installing new software is often not allowed or is infeasible.

Mainstream tools for improving accessibility have primarily looked at the first category – making access to content possible. The remainder of this chapter describes tools designed for end users to enhance collaboration and enable users who would benefit most from accessibility improvements independently address these challenges. The tools described here primarily focus on improving accessibility for blind web users, but the ideas explored can be adapted to providing accessibility improvement for people with a variety of access needs.

ACCESSIBILITY: SOCIAL ANNOTATION

Well-designed web content uses semantic annotations to assist users in browsing with a screen reader, but, for a variety of reasons, annotations are not pervasively applied. As an example, images lacking descriptions are inaccessible to screen reader users, and alternative text describing them is provided to only half [2,5] Without the quick scanning and summary afforded by a visual display, locating interesting content can be difficult. Annotations added to content can help users skip through content in a meaningful way. Heading tags (<h1> - <h6>) are a simple mechanism for conveying semantic structure and are frequently used by blind web users to navigate within a web page. Even simple annotations such as alternative text and heading tags are often not provided or used properly, and users are currently reliant on the creators of content to provide them.

The Accessmonkey framework helps to end that reliance by enabling end users to provide annotations in a shareable form. The annotations provided using Accessmonkey can benefit other users and also web developers wanting to integrate them into their own web sites [1]. IBM recently released Social Accessibility, a set of browser plugins that enable users to apply fixes to pages, and coordinate volunteers to help provide the appropriate annotations. [8] Providing these annotations can help make accessing a web site possible, but, just as in visual design, making a task possible on the web does not mean it will be an easy or intuitive experience.

Sharing annotations requires a way to address content to which each annotation applies and providing a common repository where the annotations can be accessed. Common addressing methods are XPath, pseudo-natural language descriptions, and content-specific methods, such as the MD5 hash of an image.

The addressing mechanisms that are easiest for computers to understand tend to be most difficult for

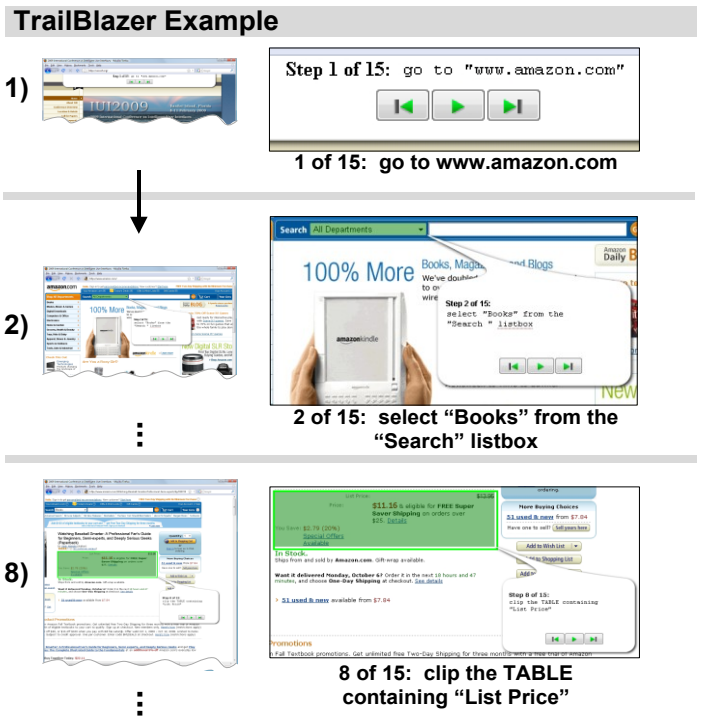

Figure 2: TrailBlazer guiding a user step-by-step through purchasing a book on Amazon. 1) The first step is to goto the Amazon.com homepage. 2) TrailBlazer directs the user to select the "Books" option from the highlighted listbox. … 8) On the product detail page, TrailBlazer directs users past the standard template material directly to the relevant product information.

people. Keyword commands uses pseudo-natural language commands to address page content and are popular in this space because they enable users to more easily understand how an annotation is changing the content they are viewing [9]. CoScripter borrows this idea of a pseudo-natural language addressing mechanism to create a wiki of how-to instructions [10]. With a number of different addressing mechanisms available, each with their own tradeoffs, it is important that these can be shared and reused in a repository that accepts them all.

The Accessibility Commons serves a unified location for annotations that allows multiple addressing types and is designed to be flexible to new types. [7] A common repository of annotations along with end user tools to help create them can help users collaboratively create web content better suited to their needs. The most straight-forward example is the user of alternative text for images that can be read by a screen reader in place of images on web pages.

The general role of annotations is to provide additional descriptions and semantics that allow end user tools to make better sense of web content.

Many different types of tools can benefit from content with more annotation. Any tool that needs to address specific content within a web page, for instance, can benefit.

USABILITY: BLAZING TRAILS THROUGH THE WEB

Applying annotations to content (such as those described in the previous section) can make access to content possible but are not usually enough to make web content usable. To be usable, users need to be able to connect individual interactions with interface components together into complete tasks. When using a non-visual interface, completing tasks on the web can be inefficient and frustrating, with each step requiring a linear search of web content to find the correct button, link, or information.

The play back components of Programming-by-Demonstration (PBD) and interactive help systems guide users through tasks step-by-step, which obviates the need for this linear search on predefined tasks. Despite the incredible potential of these tools to assist blind users, most existing systems are not usable with standard screen readers. Feedback is either indicated only visually, the mouse is required to interact with the systems, or numerous context switches between the PBD interface and the web page that is being interacted with are required.

TrailBlazer targets non-visual recording, playback and sharing of scripts (trails) to guide users through completing web-based tasks. It includes speech feedback for all interface components, provides keyboard shortcuts for all functionality, and integrates its interface directly into the web pages that are being accessed.

TrailBlazer also reuses the existing repository of CoScripts to guide users through existing how-to knowledge (Figure 1). [4] This enables blind users to immediately leverage a large existing repository of how-to knowledge. Blind users can also independently demonstrate tasks, record themselves, and then save and share the descriptions as CoScripts using TrailBlazer. These pseudo-natural language scripts bring the advantages of macro recording to a group that stands to greatly benefit.

Generalizing Trails

Blind participants in a formative study found TrailBlazer to be a great improvement over using a screen reader directly, but felt that it was too restricted because they could only use it when a script already existed for completing a task. To address this concern,

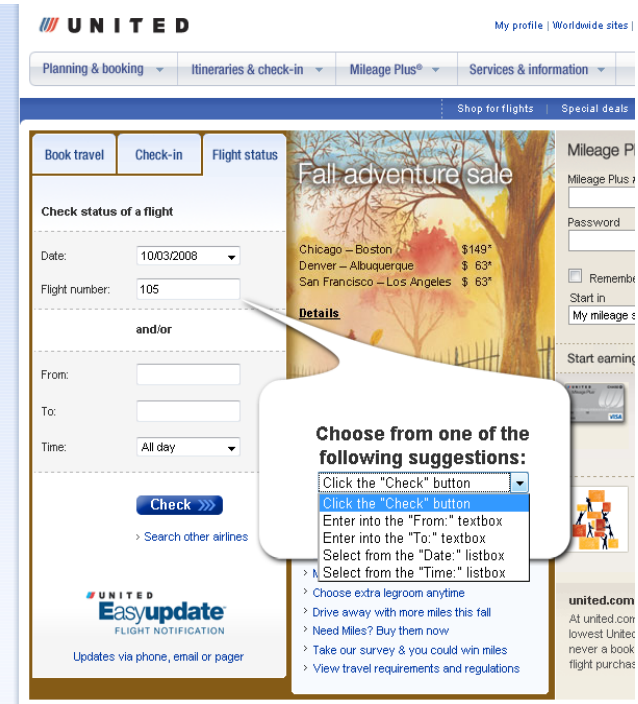

Figure 3: Suggestions are presented to users within the page context, inserted into the DOM of the web page following the element with which they last interacted. In this case, the user has just entered "105" into the "Flight Number" textbox and TrailBlazer recommends clicking on the "Check" button as its first suggestion.

we developed a novel method for suggestion-based help in order to guide blind web users through tasks, dynamically creating a new script (Figure 3). TrailBlazer creates these suggestions by combining a short, user-provided task description and an existing repository of how-to knowledge. In an evaluation of 15 user-created tasks, the correct prediction was contained within the top 5 suggestions 75.9% of the time.

Following these predictions lets users avoid lengthy linear searches in most cases. When the suggestions offered by TrailBlazer are incorrect, users only have to explore a small list of suggestions (currently 5) before completing the task as they normally would. Future research will explore how to best translate the predictions offered by TrailBlazer into improved usability – helping users more quickly complete tasks without taking away their control or causing them to be less efficient when TrailBlazer is wrong.

By guiding blind web users through web tasks the first time, TrailBlazer encourages users to create scripts that improve the efficiency of all users in the future. An on-going problem with programming-by-demonstration systems for the web is that even the

small amount of work required to define a script for a tasks makes it not worth doing for most people. The trade-off may be different for blind users for whom accessing the web is currently so inefficient. We are investigating this trade-off to see if this might make them more likely to define scripts that could then benefit everyone.

TrailBlazer currently incorporates only the knowledge from the scripts that users have explicitly recorded and shared, but exploring always-on recording to help find popular trails through web sites is an important opportunity for future work.

TrailBlazer helps users connect the individual interactions into trails that can be efficiently completed using a screen reader. This overlay on top of existing content can help make that content more efficient to access without taking away the user's control.

AVAILABILITY: BUILDING TOOLS INTO THE WEB

End user tools can dramatically improve accessibility, but people often use computers other than their own to access web content. Anyone who either requires or prefers a different interface is restricted to using only computers on which that software is already installed. In the case of the screen readers used by blind individuals, the software is incredibly expensive and not likely to be installed on most computers. Specifically, the accessibility enhancements made possible using the tools presented in the previous two sections are unlikely be available.

To address this problem, we introduced WebAnywhere, a web-based screen reader that enables blind web users to access the web from almost any computer that can produce sound without installing new software [3]. WebAnywhere works even on locked-down public terminals. To facilitate this, speech is delivered from a remote server. Pre-fetching based on a dynamic model of user behavior helps to ensure that the sounds users request to be played are likely to already be in the browser cache and perceived latency is low (Figure 4 and 5).

In addition to serving as a screen reader for the web on computers in which one is not already installed, WebAnywhere is also able to incorporate the improvements offered by Accessmonkey and the Accessibility Commons, which means the web pages it makes available are more accessible. In the future, we plan to incorporate the TrailBlazer interface into

Figure 4: WebAnywhere is a web browser that runs as a web application inside the existing web browser. It requires no special software to be installed or permissions to run, so it can provide custom interfaces wherever users happen to be.

WebAnywhere as well, and build an accessible, socially-generated version of the web into every web browser. Our public release of the system is currently being visited by approximately 700 unique users each week, providing a wealth of data that can help us both understand and improve WebAnywhere and also inform our future research directions.

Getting Tools to Users

Getting access technology and improvements that have been made by end users to the people that need them most can be difficult. Access technology is specialized software, and not installed on most computers. Locked-down public terminals prevent new software from being installed, and, for many users, the time required to install new software causes it to be more cost than it is worth. It can be difficult to overcome the cost of installing new software, which means that users may go to the trouble to benefit from the software that would be ideal for them. Access technology has a high abandonment rate, at least partially due to its complexity [6].

Users accustomed to using or reliant on a specific type of accommodation may not be able to leverage it everywhere they happen to be. WebAnywhere helps improve this cost-benefit trade-off by making loading access technology and end user improvements as easy as loading a web page.

Inaccessible content is often created because developers are not aware of the issues involved. Prior

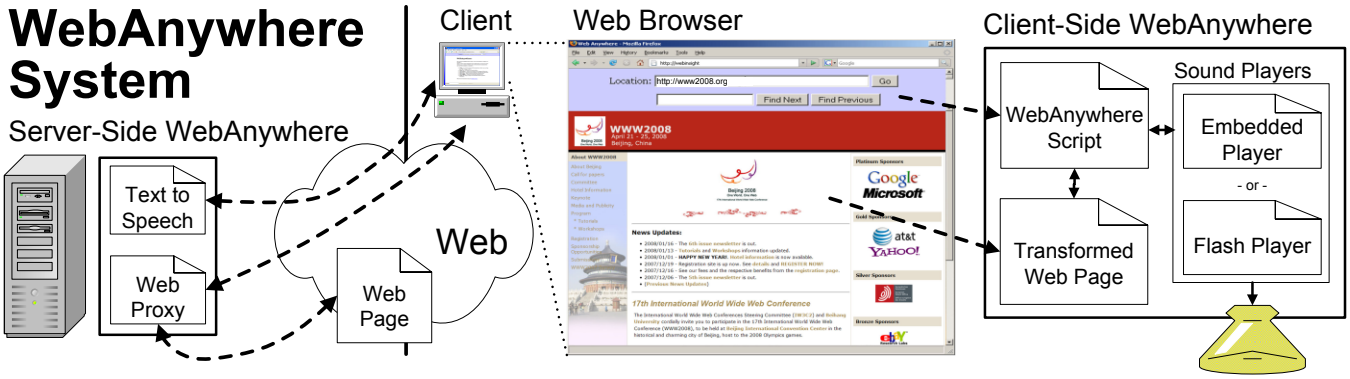

Figure 5: WebAnywhere is a web proxy is designed to provide personalized interfaces to the web to users from any computer. The interface shown adds spoken feedback and keyboard shortcuts to any web page, making any computer accessible to blind computer users. WebAnywhere is also able to deliver the accessibility improvements provided by other tools, such as Accessmonkey or Social Accessibility to any computer, making it important for helping users realize the benefit of these tools anywhere.

work has shown that web developers can be more successful at discovering accessibility issues when they view their web pages with a screen reader [11]. WebAnywhere serves as a quick way for developers to experience new interfaces.

WebAnywhere also enables users to demonstrate the problems they are having using an interface similar to TrailBlazer, capture a recording of their interaction, and then send the script off for easy review by developers. This type of end user programming can help clearly demonstrate the problems that exist in current web page, and represents a low-cost way for remote blind users to demonstrate the problems they experience using the interfaces that they use.

OPPORTUNITIES FOR FUTURE WORK

The web presents the incredible opportunity to provide everyone with efficient access to the information they need, when they need it. History has shown that it is unrealistic to expect all web developers to create content that is accessible and usable by all people. The web users who would benefit from accessibility improvements have the motivation and incentive to make their content more accessible – it is our challenge to create the tools that will allow them to do so.

Through end user tools that help users independently improve web content, we hope to enable users bypass artificial restrictions to their access in the web today and directly build in the accessibility that would be most beneficial to them. This work is part of a larger trend toward more personalized access to content that will become necessary. The web allows us to share information to an extent that we have never before experienced, but has thus far been closely tied to its visual representation. This is not working for a variety of people, using a variety of devices.

The tools described in this paper have focused on improving access to for blind web users, but can inform the design of tools for other use cases. As examples, the technology described here could directly apply to web access on both mobile phones lacking screens and small-screen devices. How we can enable authors to conveniently produce content that can be enjoyed by people with difference abilities, in difference contexts? Part of the answer is likely to create tools that enable end users to independently reshape web content to their preference.

Acknowledgements

We thank the members of the WebInSight Project at the University of Washington and CoScripter team at IBM Almaden Research Center for their input and support. We also thank the numerous participants in our user studies for their insightful comments.

REFERENCES

1. J. P. Bigham and R. E. Ladner. Accessmonkey: A Collaborative Scripting Framework for Web Users and Developers. In *Proceedings of the International Cross-Disciplinary Conference on Web Accessibility (W4A 2007)*, pages 25-34. Banff, Alberta, Canada, 2007.
2. J. P. Bigham, A. C. Cavender, J. T. Brudvik, J. O. Wobbrock, and R. E. Ladner. WebinSitu: A Comparative Analysis of Blind and Sighted Browsing Behavior. In *Proceedings of the 9th International ACM Conference on Computers & Accessibility (ASSETS 2007)*, pages 51-58. Tempe, Arizona, USA, 2007.
3. J. P. Bigham, C. M. Prince and R. E. Ladner. WebAnywhere: A Screen Reader On-the-Go. In *Proceedings of the International Cross-Disciplinary Conference on Web Accessibility (W4A 2008)*, pages 73-82. Beijing, China, 2008.

4. J. P. Bigham, T. Lau and J. Nichols. TrailBlazer: Enabling Blind Users to Blaze Trails Through the Web. In *Proceedings of the 12th International Conference on Intelligent User Interfaces (IUI 2008)*. Sanibel Island, Florida, USA, 2009.

5. J. P. Bigham, R. S. Kaminsky, R. E. Ladner, O. M. Danielsson and G. L. Hempton. WebInSight: Making Web Images Accessible. In *Proceedings of the 8th International ACM Conference on Computers and Accessibility (ASSETS 2006)*, pages 181-188. Portland, Oregon, USA, 2006.

6. M. Dawe. Desperately Seeking Simplicity: How Young Adults with Cognitive Disabilities and Their Families Adopt Assistive Technologies. In *Proceedings of the SIGCHI conference on Human Factors in computing systems (CHI 2006)*, pages 1143-1152. Montreal, Canada, 2006.

7. T. Hironobu, S. Kawanaka, M. Kobayashi, T. Itoh and C. Asakawa. Social accessibility: achieving accessibility through collaborative metadata authoring. In *Proceedings of the 10th International ACM Conference on Computers and Accessibility (ASSETS 2008)*, pages 193-200. Halifax, Nova Scotia, Canada, 2008.

8. S. Kawanaka, Y. Borodin, J. P. Bigham, D. Lunn, H. Takagi and C. Asakawa. Accessibility Commons: a Metadata Infrastructure for Web Accessibility. In *Proceedings of the 10th International ACM Conference on Computers and Accessibility (ASSETS 2008)*, pages 153-160. Halifax, Nova Scotia, Canada, 2008.

9. G. Little and R. Miller. Translating keyword commands into executable code. In *Proceedings of the 19th Annual Symposium on User Interface Software and Technology (UIST 2006),* pages 135-144. Montreux, Switzerland, 2006.

10. G. Little, T. Lau, A. Cypher, J. Lin, E. M. Haber, E. Kandogan. Koala: capture, share, automate, personalize business processes on the web. In *Proceedings of the SIGCHI conference on Human factors in computing systems(CHI 2007)*, pages 943-946. San Jose, California, 2007.

11. J. Mankoff, H. Fait, and T. Tran. Is your web page accessible? A comparative study of methods for assessing web page accessibility for the blind. In *Proceedings of the SIGCHI conference on Human factors in computing systems (CHI 2005), pages 41-50.* Portland, Oregon, USA, 2005.