Bells, Whistles, and Alarms: HCI Lessons Using AJAX for a Page-turning Web Application

Juliet L. Hardesty
User Interface Design Specialist
Digital Library Program
Indiana University
Herman B Wells Library, W501
1320 E. 10th Street
Bloomington, IN 47405 USA
jlhardes@indiana.edu

CHI 2011, May 7–12, 2011, Vancouver, BC, Canada.
ACM 978-1-4503-0268-5/11/05.

Abstract
This case study describes creating a version of METS Navigator, a page-turning web application for multi-part digital objects, using an AJAX library with user interface components. The design for this version created problems for customized user interactions and accessibility problems for users, including those using assistive technologies and mobile devices. A review of the literature considers AJAX, accessibility, and universal usability and possible steps to take moving forward to correct these problems in METS Navigator.

Keywords
AJAX, accessibility, universal usability

ACM Classification Keywords
H.5.2. Information interfaces and presentation: User interfaces - *user-centered design, standardization.*

General Terms
Design, human factors, standardization

Introduction
AJAX (Asynchronous JavaScript and XML) is a widely used method for developing Web 2.0 applications

(called Rich Internet Applications, or RIA's), both to enhance certain features and interactions on a web site as well as for interactive user interface components. A variety of AJAX libraries exist with different types of user interface components that allow for new or more dynamic ways of interacting with content on the web (see Dojo, Ext JS, GWT, jQuery, script.aculo.us, and YUI for just a few examples of the many AJAX libraries offering user interface components). The following case study considers a page-turning web application, METS Navigator, developed by the Indiana University Digital Library Program. Open-source libraries from Ext JS[4] were used to create the latest version of this web application, meant to enhance portability and ease of use by implementers and end-users.

Using AJAX creates dynamic HTML (HTML produced by JavaScript at the time the page is rendered in the browser), which is not viewable in the source code of a web page without the use of special tools. This dynamic HTML is, however, used by screen reading software and other assistive technologies when interpreting the content of a web page. When evaluating the code created by the AJAX libraries used in METS Navigator and addressing accessibility issues to ensure that METS Navigator was, indeed, usable by everyone, problems with dynamic HTML, accessibility, and mobile device usability were uncovered. From this experience, lessons learned and tips to avoid similar problems are shared for the benefit of the human-computer interaction (HCI) community.

Literature Review

In reviewing the literature on AJAX and accessibility, problems of accessible use seem inevitable. Discussing in 2007 how a standardized approach is needed to incorporate the Semantic Web into Web 2.0 development, Cooper acknowledges that "Web 2.0 is an emergent phenomenon rather than a specific technology or set of technologies" and, thus, "there is no single designer or responsible authority that ensures that the technologies and practices support accessibility."[3] Likewise, Lembree discusses in 2008 and again in 2010 specific problems in making AJAX accessible: JavaScript is required, web conventions such as the Back button and bookmarking are broken when using AJAX, content changes go unnoticed by users, keyboard access and focus can be lost, and search engine indexing can be diminished, making content even more difficult to discover in the first place.[15][16]

Preceding the call for a standardized approach to Web 2.0 development, Ben Schneiderman in 2000 expands on the concept of universal access from the Communications Act of 1934 (covering telephone, telegraph, and radio services) to discuss the idea of universal usability, or making the web available and accessible for 90% of households. The difficulties inherent with this approach, however, include technology variety, user diversity, and gaps in user knowledge.[21] These issues are still core problems encountered with AJAX and accessibility. Different technologies have different capabilities for using JavaScript and showing graphical user interfaces. Users approach these interfaces from various perspectives and with a range of expectations based on their previous experiences online.

A recent survey of IBM developers, whose company touts their own accessibility department, internal guidelines for accessible web development, and

assistive technologies for testing, identified one of the biggest difficulties in making online products accessible when using AJAX as "finding workarounds to problems with toolkits, cross-browser support, and third party components."[23] Gibson writes in 2007 about the need for AJAX toolkits to include accessibility, such as landmarks, roles, and states prescribed by WAI-ARIA, by default. "The first step is to enable toolkits with accessibility, then applications built using these toolkits will inherit accessibility."[6] Insisting on a positive outlook to the problem, however, Gibson uses the perspective of history to state that similar to when the web was first created, Web 2.0 has not been accessible from its start and requires specific tools (like a mouse) and different implementations for different browsers. But as development continues, standards will win out and accessibility will improve.

Until the day arrives when standard AJAX development incorporates accessibility, suggestions abound for best development methods and tools to aid in creating accessible AJAX web applications. More than one suggestion revolves around marking up content in a semantic manner as a base from which to develop.[3][6][8][22] A development technique promoted early on in the AJAX timeline by Jeremy Keith is call Hijax.[12] The concept prescribes developing the web application without using any JavaScript (incorporating pages that fully refresh) and marking up the content as semantically as possible, then intercepting (hijacking) calls to the server that would fully refresh the page and using AJAX to enhance the behavior of the user interface. This method is also known as progressive enhancement. Using user interface toolkits from AJAX libraries makes this method ineffectual, however, since beginning development without JavaScript requires creating an entire user interface that then goes away when the AJAX user interface component is applied, extending development time and complicating user interaction design.

Tools developed for use with Rich Internet Applications have also been suggested as ways to improve accessibility. Google-AxsJAX uses WAI-ARIA to "inject" accessibility based on the document object model (DOM) structure.[20] WIMWAT is a more recently developed tool that proposes evaluating a web application to see if it contains any RIA widgets (such as a carousel slideshow widget). This data can then be used by developers to evaluate the accessibility of those widgets or by screen reading software to enhance the information provided about content and content changes to screen reader users.[2]

Expectations for interactions in web applications have increased since the introduction of AJAX development techniques. Toolkit components from AJAX libraries that provide functionality in ready-made interfaces only requiring content have furthered the speed at which enhanced interaction has become commonplace. In the long run, however, the underlying structure and interaction methodology assumed by these developments (mouse interaction and full-scale visuals of the entire interface) will not withstand real life and the needs of real users. A look at two studies of mobile phone users and screen reader users from 2006 and 2009 respectively, shows that traditional interactions with web content do not work with Web 2.0 applications.[18][7] User interaction designers need to reconsider what users need from more interactive web applications and those needs have to include a more

varied audience (visual variety, hearing variety, and age-related variety) among users and devices. Design on the web no longer revolves around the desktop browser and the user with a keyboard and a mouse. The following case study highlights this miscalculation and discusses approaches to recover from these development missteps.

METS Navigator History

METS Navigator was created in 2004 by the Digital Library Program (DLP) at Indiana University as a solution for delivering scanned images of pages and other multi-part digitized objects in a web interface, allowing end-users to view all digitized images of an item as a complete object. Page-turning web applications were not well developed at that time and the needs of the Digital Library Program included use of the XML metadata standard, Metadata Encoding and Transmission Standard (METS), to allow navigating the structure of the digitized object.[11]

Early versions of METS Navigator were written in Java Struts and incorporated XML data to construct the navigation. The needs of users, however, were not being fully met by this implementation. Basic browsing was possible per digital object (e.g., paging through a book), but each time a page was "turned," the browser was completely reloaded, disrupting the feel of a page turning action. Additionally, browsing across groups of objects and searching within an object was not possible. Discovering this digital content was also not always straightforward. Some items served via METS Navigator are not part of any formal collection and can only be accessed through Indiana University's online library catalog (IUCAT). Providing a way for these individual items to be broadcast by interested parties without requiring the use of Digital Library Program development and hosting time would increase discoverability and create a way for the Digital Library Program to serve content that anyone can use.

METS Navigator: The AJAX Version

The early Java Struts/XML web application version of METS Navigator was released as open source software in 2006, but the time necessary to create documentation and release updates to the open source version was scarce. Particularly since the Digital Library Program faced time and resource limitations, it seemed more likely that other web sites would want to easily serve Digital Library Program content than that an entirely separate institution or group of developers would wish to install, use, and develop on top of the Java codebase for METS Navigator. Based on the need to increase the availability of the Digital Library Program's digitized objects by letting others use METS Navigator and in search of a way to better fulfill user needs for a more authentic page turner experience with searching capabilities, a new design model envisioning METS Navigator as a hosted web service came into being.

Thus, the most recent version of METS Navigator was developed using AJAX to support cross-collection browsing (e.g., issues of a journal), document-centric browsing (e.g., structured table of contents), and the ability to "turn pages" without a complete refresh of the browser window. In addition, digitized items with available optical character recognition (OCR) conversions – full text generated from the digitized image - are searchable across the entire collection in a single AJAX interface. These options are configurable

Figure 1. METS Navigator – AJAX version on desktop browser, showing digitized issues of Outdoor Indiana Magazine from 1934-1993 with browsing and searching capabilities in addition to page-turner features.

and the interface flexible enough to show or hide these features, depending on the collection. This version is also compatible with any third party web site or web application. With a few calls to hosted JavaScript and Cascading Style Sheet (CSS) files along with a JavaScript function to set up the previously mentioned configurations, any site internal or external to the Digital Library Program can serve the DLP's digitized content, providing wider access and distribution capabilities for these special collections previously unavailable for even in-person viewing.

For this version of METS Navigator, AJAX libraries provided a way to easily create user interface components and allowed development time to focus on the delivery of content and the search capabilities of METS Navigator. At the time of development there were a variety of AJAX libraries available, but the available user interface components varied between the different libraries. Ext JS, now part of a platform called Sencha[4], provided all of the user interface components necessary to build the page-turner interface: tabs, panes, general paging controls, a

search interface that allowed paging through search results, a tree menu, and customization capabilities.

The Digital Library Program hosts the Java Struts code that interprets the metadata for items and collections and the user interface is offered via a set of JavaScript files. Configuration to control the size, layout, and features of the page-turner viewer (for example, if searching capabilities are included or not) is handled by JavaScript functions. A persistent URL passed as a parameter on the URL for the web page gathers the XML data necessary to show the navigational structure and page images. The use case that best exemplifies the Digital Library Program's goal for this version is to enable Indiana University's online library catalog (IUCAT) to serve digitized content directly within the online catalog record, removing the need for the end-user to leave IUCAT to retrieve these electronic resources. Now, by incorporating a set of JavaScript files and functions and a persistent URL as a parameter on a web site's URL, METS Navigator has the capability to be used anywhere on the web to deliver digitized content from the Digital Library Program.

Some usability issues related to using AJAX in web applications were recognized and included in the project requirements for this version. Bookmarking and the browser Back button were noted AJAX-related issues that needed to be resolved if this version were to move forward successfully. Developers did the research and work involved to correct these issues and worked out the following solutions. For bookmarking, the developers added a page parameter to the URL for the web page via JavaScript each time a page image was loaded so any page within any item could be retrieved from a bookmarked URL. The browser's Back button was re-enabled through the use of Ext JS's History functions, making it possible to write each loaded page of an item to the browser's history.

Lack of discoverability via search engines is another issue common to AJAX web applications, as content loaded within a page that never reloads tends to never be found by search engines indexing the static pages of a web site. Within the Digital Library Program team working on METS Navigator, there is both a desire to make METS Navigator content more discoverable via search engines as well as a question about the necessity of such a feature. Some content viewed via METS Navigator exists more or less on its own with no other online information or context. These are items that have been digitized in the past but are not connected to any broader collection or set of items that can provide context. These items would currently benefit from discoverability via search engines. Many items currently viewed using METS Navigator, however, are already part of other web sites being indexed by search engines. These sites contain additional context so METS Navigator items are better served through discovery via those collections rather than as standalone items online. In addition, development is moving towards an embeddable version of METS Navigator, meaning the page-turner and the items it shows will always be viewable within the context of a web site (internal or external to the Digital Library Program), raising further questions about what is really worthwhile to expose for search engine indexing at this stage of development.

The AJAX-related usability issues previously described were considered early on, managed with some amount of developer effort, and included enhancements beyond

the use of Ext JS. Other usability issues requiring additional work, however, were unforeseen and even reached beyond the project's scope.

User Interface Issues

Problem: Customizing and Validating HTML/CSS

Customizing the HTML markup and Cascading Style Sheets (CSS) was the first unforeseen user interface problem encountered. In trying to customize the look and feel of the different user interface components, concerns arose that different customizations were not displaying the same across different browsers. Additionally, customizing the dynamic HTML output was a difficult task since this output was generally only available to the designer within compiled JavaScript files, files that are not meant for editing. These explorations lead to trying to view the HTML source produced by the AJAX libraries. Viewing the dynamic HTML was possible using a tool such as the Firebug plug-in for Firefox[5], but validating that HTML source from Firebug was not possible and copying and pasting to validate was as time-consuming and undesirable as editing the compiled JavaScript files. The dynamic HTML produced by the AJAX libraries was often cumbersome and definitely not semantic (i.e., HTML tables used for layout or several layers of HTML div's, nested within each other). Especially after combining several user interface components such as the panes and a tree menu, the dynamic HTML produced did not validate.

These problems should not be considered as anything negative towards Ext JS – the focus of most AJAX libraries is to provide functionality in stable JavaScript that performs the same across browsers and platforms. The HTML dynamically produced by these components is a side product, not meant for editing or even viewing. The error in this approach is that the product created does not meet standards or provide a flexible enough user experience to match audience needs. Just as the surveyed IBM developers noted, the Digital Library Program found that dealing with problems in AJAX libraries is particularly challenging.

Solution: Customize as Much as Possible

Relying on AJAX libraries to create the HTML means just that – using the libraries. Customizing the look and feel of the user interface from these libraries is possible to a point. Larger button images and tool tip text were put in place on the Page Turner Toolbar using custom JavaScript functions from that Ext JS library and background images styled within CSS. These changes made the controls for turning the pages of an item easier to use and understand.

The project scope of METS Navigator, however, has not allowed time for contributing changes to Ext JS's AJAX libraries to improve the dynamic HTML. Becoming part of Ext JS open-source development appears to be the only real way to change the HTML output from its libraries. Future development might require this kind of work, but for now, METS Navigator currently produces invalid HTML, impacting efforts to improve accessibility.

Problem: METS Navigator Inaccessible

Fairly soon after discovering that viewing and validating the dynamic HTML was a problem, the accessibility of METS Navigator as an AJAX web application came into question. If the dynamic HTML was not written to standards, the likelihood that assistive technologies like screen readers could interpret METS Navigator content

was low. The Adaptive Technology & Accessibility Centers at Indiana University[9] tested METS Navigator using screen reading software and confirmed that, indeed, the content and page-turning controls were not accessible and the tree menu and search interface only provided limited access. Using user interface components from an AJAX library as a means of building an entire interface was limiting access for users of screen reading software and keyboard-only users (users who do not use a mouse).

Accessibility of AJAX user interface components has not been a development priority among the community. Certain libraries, such as YUI and GWT, allow for adding accessibility by way of WAI-ARIA landmarks, roles, and states, which can be used by screen readers and other assistive technologies to signal changes in information on a page using JavaScript[24], but by default AJAX libraries do not produce components with this kind of usability in mind. Testing for accessibility is easy enough in terms of keyboard access (using a web page, widget, or web application without a mouse is enough to test keyboard access). Screen reader testing requires special equipment and skills to know how to use the software. There are some tools available, such as the Fangs plug-in for Firefox[13], which provide a limited simulation of how a screen reader would interpret and read a web page. These tests can be quite revealing to developers and designers both and can be helpful when determining whether or not an AJAX user interface component is worth implementing.

Solution: Customization for Keyboard Access
Fortunately, through customization capabilities within the Ext JS libraries along with other JavaScript and CSS techniques, METS Navigator developers and designers improved keyboard access. A KeyMap function from Ext JS allowed keys to be assigned for access to the "Browse Collection" and "Table of Contents" tabs inside the Navigation pane on the left side of the application (certain collections show both tabs while others only use the "Browse Collection" tab). Further JavaScript commands assigned the use of the ENTER key to submit a search and the TAB key to move focus to search results in the Search pane on the right side of the application. CSS changes created a border around links and buttons of focus within METS Navigator as well, helping keyboard users visually track browser focus within the page-turner. The effort to implement keyboard accessibility was not insignificant, requiring research into keyboard controls available in Ext JS, creating a plan for overall keyboard access, and testing the implementation in various browsers and platforms to ensure comparable behavior. The end result, however, was an overall improvement in functionality for most users, since clicking the mouse was no longer required to conduct a search and browser focus was always visible.

The keyboard interactions generally involve the TAB key, ARROW keys, and ENTER key (keys commonly used across browsers for navigating and working with content) so they lend themselves to discoverability but there are differences in keyboard interaction among the sections of the page-turner. Due to the different keyboard interactions required to move through the tree menu versus the search results and some custom keyboard assignments, the Help page for METS Navigator now includes a section on keyboard accessibility.[17] These changes do nothing to help screen reader access but they are an accessibility improvement, nonetheless.

Problem: Interface Unusable on Mobile Devices
METS Navigator with an AJAX interface is not usable on mobile devices, particularly mobile phones with small screens. The controls become too small for touch interfaces and the interface resizes to fit the size of the screen, meaning METS Navigator is often times a top bar of controls and not much else.

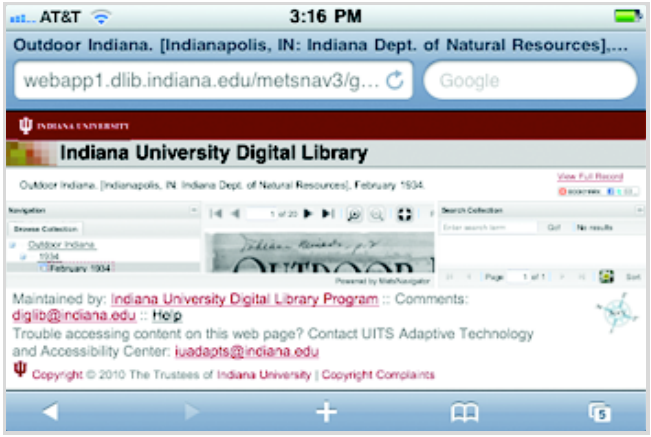

Figure 2. METS Navigator interface as displayed in landscape mode (horizontally, for largest view) on iPhone.

METS Navigator on an iPad or tablet-style device is a better experience than a touch-screen phone, but the interface is still limited by the width of the screen and page images are not fully viewable with the Navigation and Search columns showing (See Figure 3). The columns and controls are easier to see on the larger screen of a tablet and those columns are collapsible by the user so the experience is not as limiting, but the interface was not constructed with such a small screen in mind. This oversight needs to be addressed when moving forward with future METS Navigator development.

Solution: Pending
There are options for addressing the current interface problems on mobile devices: use media queries to arrange a CSS that changes the banner and footer areas of the page so more of METS Navigator can show on small screen devices; try modifying the CSS used by the AJAX libraries in small screen devices to increase the size of controls or define a different height and layout for the entire interface; or develop a completely separate METS Navigator interface for use with small screen devices.[1]

Considering the options described, using media queries to modify the CSS rendered based on screen size seems like the best solution. The styles for the information surrounding the page-turner (header, footer, and bibliographic content about the collection) are easily manipulated. CSS can be used to control the look and feel of the AJAX components to a point but not every function or component can be styled by a CSS class or identifier, so limits may be encountered when reworking the AJAX user interface for mobile devices. Creating two separate METS Navigator applications is not a sustainable development plan, however, so working with the CSS to change the display of METS Navigator based on the screen size of the device being used seems like an option that could work.

Currently, METS Navigator uses an Ext JS function called Viewport that assigns components to regions

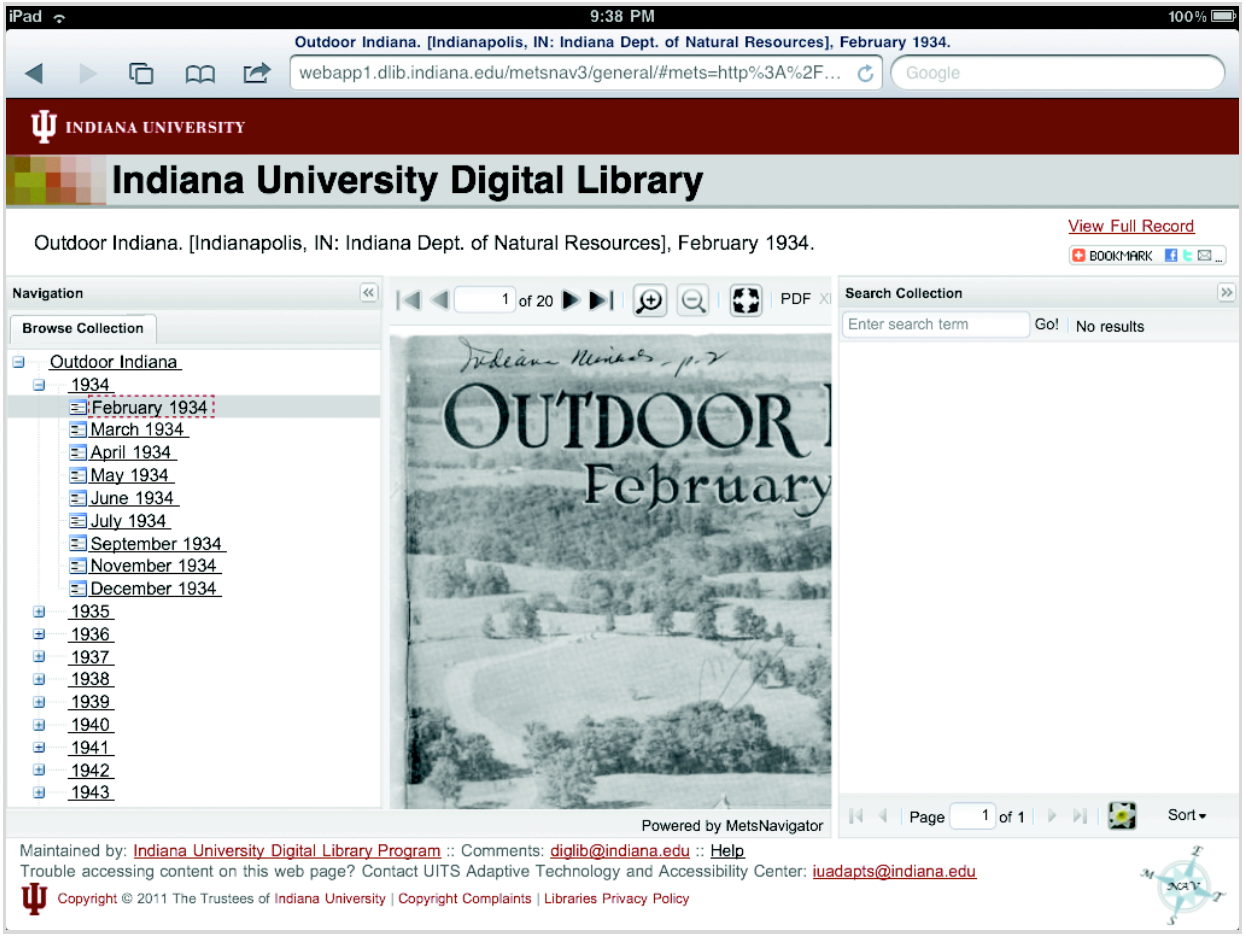

Figure 3. METS Navigator as displayed in landscape mode on iPad.

(north, south, east, west, or center) and specifies the widths of those components. In the rendered HTML, this translates to inline styles of absolute positioning (i.e., style="left:0px; right:0px;") and width (i.e., style="width:300px;"). If an external CSS file called by a media query can override these inline styles (using the "!important" statement, for instance) then positioning can be changed on a smaller screen size to a vertical rather than horizontal layout and width can be adjusted to 100%. These changes might work for

mobile devices, particularly mobile phones. Tablet-sized interfaces might only require a change to the width of the side columns, which can also be accomplished via external CSS styles. Unfortunately, using the "!important" statement in CSS styles can introduce problems, since it disrupts the cascading rules of CSS and can complicate maintenance and development work.[14] But this option is viable at this time using the current Ext JS libraries and more maintainable than creating a separate METS Navigator application for mobile devices.

These explorations to improve METS Navigator on mobile devices have yet to occur. Creating mobile device styles that will work when METS Navigator is embedded in a third party web site will present even more challenges. As development continues, however, mobile access will need to be addressed along with accessibility and standards-compliant HTML.

Future Development

The next version of METS Navigator will attempt to implement the embedded page-turner – it will be used in a separate Digital Library Program application (IU's Finding Aids site[10]) to show digitized items available within full-text finding aids of various archives and special collections at Indiana University. See [19] for an example of a finding aid (a guide to a physical archival collection) that currently contains digitized items for viewing through an earlier version of METS Navigator. The embedded version will test the portability and hosting model that was one of the reasons for moving to a user interface constructed via AJAX.

Development has progressed enough using the Ext JS libraries that re-writing METS Navigator using a different AJAX library would involve too much time for this next release. Ext JS is working on components that are accessible or incorporate WAI-ARIA (see Key Feed Viewer and ARIA Tree examples under "Accessibility Samples" in [4]) but they are labeled as "Experimental" and are not the standard way of writing user interface components in this AJAX library. It also remains unclear whether or not any of the user interface components have been improved to produce valid dynamic HTML that could be semantically interpreted by screen readers.

In addition, due to the fact that the content currently shown in METS Navigator is only digitized images (.tiff, .jpg, or .gif files), true accessibility has never been possible. By nature, image content has limited accessibility for screen reader users and other assistive technologies. In the future, however, METS Navigator will be the content delivery mechanism not only for digitized images but also full text, video, and audio content, much of which can be interpreted by any user with the right technology. The Digital Library Program will have a responsibility to ensure that this content can be used by everyone, including screen reader users and mobile device users, in order for METS Navigator to remain a viable and usable mode of content delivery.

Conclusion

Using AJAX libraries to create a portable interactive user interface has to be weighed against the current costs of limited accessibility and a diminished overall quality in the final product. METS Navigator uses AJAX libraries to create its user interface, enabling page turning, internal navigation, and searching without a web page reload and the ability to embed Digital Library Program content in any web site. However,

METS Navigator also suffers from a lack of screen reader accessibility and poor usability on mobile devices.

Standardized methods for developing AJAX web applications that include accessibility are on the horizon for AJAX toolkits, but these practices are not yet commonplace. Customizations to the AJAX libraries used in METS Navigator aid accessibility for keyboard-only users but problems persist for screen reader users. Additionally, plans to provide full text via METS Navigator will be hampered if the application cannot be made accessible to all users. Currently, only page images are served via METS Navigator, limiting the usefulness of the content for screen reader users, but using this application to serve full text content will require that it work for everyone, including those using assistive technologies and mobile devices. CSS changes might improve the experience of using METS Navigator on mobile devices, but since the AJAX libraries in use currently render some styles inline, changes will have to be forced outside of CSS rules and could complicate future development work.

Improving the rendered HTML by contributing changes to Ext JS's JavaScript libraries might be the next step to make METS Navigator accessible for users of screen readers and mobile devices. This option will need to be compared to the time and cost of re-writing METS Navigator using a different AJAX library to determine the best route to complete accessibility. This work is not occurring for METS Navigator's next release, but will need to happen before METS Navigator begins delivering full text content and other media. Accessible usability is a model by which all web content should be developed but project goals and technology limits can impede these larger objectives. For now, the hope is that the current state of METS Navigator is as universally accessible as possible.

Acknowledgements

I would like to thank Michelle Dalmau, David Jiao, Jenn Riley, and Cliff Ingham for their helpful feedback in preparation of this case study. I would also like to thank the Adaptive Technology & Accessibility Centers at Indiana University for their evaluations of METS Navigator using assistive technologies.

Citations

[1] Andrew, R. How to use CSS3 media queries to create a mobile version of your website. *Smashing Magazine.* http://www.smashingmagazine.com/2010/07/19/how-to-use-css3-media-queries-to-create-a-mobile-version-of-your-website.

[2] Chen, A. Widget identification and modification for Web 2.0 access technologies (WIMWAT). *SIGACCESS Accessibility and Computing*, 96, ACM Press (2010), 11-18.

[3] Cooper, M. Accessibility of emerging rich web technologies: Web 2.0 and the Semantic web. *Proc. W4A 2007,* ACM Press (2007), 93-98.

[4] Ext JS. http://www.sencha.com/products/js/.

[5] Firebug. http://getfirebug.com/.

[6] Gibson, B. Enabling an accessible Web 2.0. *Proc. W4A 2007,* ACM Press (2007), 1-6.

[7] Hailpern, J., Reid, L.G., Boardman, R., and Annam, S. WEB 2.0: Blind to an accessible world. *Proc. WWW 2009,* ACM Press (2009), 821-830.

[8] Horton, S. Designing beneath the surface of the web. *Proc. W4A 2006,* ACM Press (2006), 1-5.

[9] Indiana University Adaptive Technology & Accessibility Centers. http://www.indiana.edu/~iuadapts/.

[10] Indiana University Digital Library Program. Finding Aids. http://www.dlib.indiana.edu/collections/findingaids.

[11] Indiana University Digital Library Program. METS Navigator. http://metsnavigator.sourceforge.net/.

[12] Keith, J. Hijax: Progressive enhancement with Ajax. *XTech 2006,* (2006). http://domscripting.com/presentations/xtech2006/.

[13] Krantz, P. Fangs – the screen reader emulator. http://www.standards-schmandards.com/projects/fangs/.

[14] Lazaris, Louis. !important CSS Declarations: How and When to Use Them. *Smashing Magazine.* http://www.smashingmagazine.com/2010/11/02/the-important-css-declaration-how-and-when-to-use-it/.

[15] Lembree, D. Ajax and web accessibility. *Accessing Higher Ground: Accessible Media, Web, and Technology Conference,* (2008). https://docs.google.com/present/view?id=dc39pw4c_67hcjvc3dt.

[16] Lembree, D. Making JavaScript Accessible. *BayJax – the Bay Area Ajax and JavaScript Meetup,* (2010). http://www.slideshare.net/webaxe/making-javascript-accessible.

[17] METS Navigator Help. http://webapp1.dlib.indiana.edu/metsnav3/help.html.

[18] Plos, O. and Buisine, S. Universal design for mobile phones: a case study. *Ext. Abstracts CHI 2006.* ACM Press (2006), 1229-1234.

[19] President Herman B Wells' speeches, 1937-1962. http://purl.dlib.indiana.edu/iudl/findingaids/archives/InU-Ar-VAA2642.

[20] Raman, T.V. Cloud computing and equal access for all. *Proc. W4A 2008,* ACM Press (2008), 1-4.

[21] Schneiderman, B. Universal Usability. *Communications of the ACM 45,* 5 (2000), 84-91.

[22] Shelly, C. and Young, G. Accessibility for simple to moderate-complexity DHTML web sites. *Proc. W4A 2007,* ACM Press (2007), 65-73.

[23] Trewin, S., Cragun, B., Swart, C., Brezin, J., and Richards, J. Accessibility challenges and tool features: An IBM web developer perspective. *Proc. W4A 2010,* ACM Press (2010).

[24] W3C Web Accessibility Initiative. WAI-ARIA Overview. http://www.w3.org/WAI/intro/aria.