Enabling an Accessible Web 2.0

Becky Gibson
IBM Emerging Technologies
5 Technology Park Drive
Westford, MA USA
01 978 392-6101
gibsonb@us.ibm.com

ABSTRACT

The next generation of the Web is relying on new technologies to build rich interfaces and applications which enable community, collaboration, social networking and enhanced interactions. This has implication for people with disabilities who have come to rely on the Web to provide more independence, work opportunities, and social interactions. New specifications such as Accessible Rich Internet Applications (ARIA) are being developed which provide more semantic information about Web components and can enable enhanced accessibility. In addition, toolkits and testing tools are making it easier to reach the nirvana of accessibility by default in Web 2.0 projects.

Categories and Subject Descriptors

H.5.4 [**Information Interfaces and Presentation**]: Hypertext/Hypermedia – *Navigation, User Issues.* H.5.2 [**Information Interfaces and Presentation**]: User Interfaces – *Graphical User Interface, Interaction Styles.* I.7.2 [**Document and Text Processing**]: Document Preparation – *Hpertext/Hypermedia, Markup languages, Scripting languages.*

General Terms

Design, Human Factors, Standardization, Languages

Keywords

Accessibility, ARIA, HTML, DHTML, JavaScript, Web 2.0

1. INTRODUCTION

The Web is constantly evolving and changing. In the beginning the Web was a click, wait, replace model. A user would type a Web address, and a page was loaded. To interact or get more details about the current topic the user would provide information via a form, click submit and wait for a new page to be downloaded. The foundation of the Web was to be able to share information. Until competent search engines were developed, the user would click on provided links to traverse the myriad of data. The Web opened up a vast world of knowledge to people. Except it was, and still is, very visually oriented and relies on the mouse interface to navigate.

Eventually the Web became more accessible. Mechanisms are available and supported in Web browsers to provide keyboard navigation and to enable assistive technologies to allow persons with disabilities to use the Web. The Web Content Accessibility Guidelines 1.0 were developed in the W3C to provide guidance and techniques to make the Web Accessible [1]. The original Web was not perfect but with some work by Web developers it could be used by all.

Now we are entering the next generation of the Web – Web 2.0. It is all about interaction, collaboration, and social networking. It is more dynamic and fluid and no longer just static pages of information. This presents yet another challenge for people with disabilities. These new interaction models are pushing the limits of the technologies of the Web and the ability of assistive technologies to interpret the changing face of the Web.

This paper describes new technologies and tools being developed to help enable accessibility for Web 2.0. The Accessible Rich Internet Applications specification is adding semantic metadata to the new rich user interface components being created on the Web. Updates to accessibility application programming interfaces provide the mechanism to communicate the advanced Web 2.0 features to assistive technologies. Toolkits make developing Web 2.0 faster and easier. And testing tools are advancing to assist with the development and run-time testing of Web 2.0 applications.

2. WHAT IS WEB 2.0?

The term “Web 2.0” was coined by O’Reilly Media at a conference in 2004 [2] and it has become the mechanism to refer to the next generation Web. Rather than just a static repository for data, the Web has become a platform for applications and the enabler for on-line participation, collaboration, harnessing collective intelligence [2] and more. The key concepts are participation and dynamic interaction.

2.1 Web 2.0 Technologies

The most widely used technologies beyond basic HTML for implementing Web 2.0 are scripting and Cascading Style Sheets (CSS). The use of JavaScript to make Web sites more dynamic has been growing steadily and CSS usage has been increasing even faster. Today, 59% of sites use JavaScript and 54% use CSS. This is up from 37% and 13% respectively in 2001 [3]. CSS enables richly styled elements which can be explicitly places on the page. Scripting provides the programming mechanism to update styles, \perform calculations, add logic, validate data on the client and dynamically update the page via XmlHttpRequest.

XmlHttpRequest (XHR) is an application programming interface that can be used via JavaScript to transfer data over the standard Web Http protocol to update portions of the page. The initial implementation provided for transferring data via eXtended Markup Language (XML) but other forms of data are now common. Scripting and data transfer via XmlHttpRequest are the original key technologies in the term Ajax – Asynchronous JavaScript and XML. Ajax refers to dynamically updating only portions of a page rather than the traditional Web 1.0 model of requesting and reloading entire pages at a time.

With high speed internet connections becoming the norm, multimedia is also becoming a big part of Web 2.0. Sites are embedding video and sound, as well as using Adobe Flash to create multimedia experiences. There are still hurdles to overcome to provide captioning for live media presentations and make easy captioning and transcription a reality. There are documented techniques for making Flash more accessible [4].

2.2 Web 2.0 Implications

Web 2.0 technologies have changed the way the Web is used and perceived. Rather than a mechanism to provide information, the web is now interactive and harnessing the wisdom of many through wikis, blogs, and communities. New terms have been coined or resurrected to explain the new phenomenon; crowdsourcing [5], social networking, collective intelligence and more. Companies no longer only use the Web as a tool for information dissemination and marketing but as a way to include the customer base in design, development and support. Web 2.0 is about inclusion, harnessing the wisdom of many to reach new conclusions and optimizing research and learning. There are even virtual communities such as Second Life and World of Warcraft Games where people can assume new personalities and build an on-line reputation.

As the interactions get more complex, the user interfaces are also becoming more rich and interactive. No longer can a site use simple lists of links for navigation. The increased complexity of sites are requiring more sophisticated user interface elements similar to those of the desktop such as tree controls, tabbed interfaces, floating dialogs, and toolbars. Users are no longer satisfied to enter simple text in an HTML textarea element but want to create styled, rich text when creating comments, emails, and posts within social networking sites. This has implications for accessibility and access by persons with disabilities.

2.3 Web 2.0 Accessibility Concerns

The Web has opened up many opportunities for people with disabilities. People with disabilities rely on the web for everyday tasks as well as for employment, learning and entertainment [6]. On-line shopping allows people with visual or mobility impairments to shop independently without traveling to a physical store location or requiring assistance from others. Learning opportunities delivered via the web offer further education for people from all walks of life and abilities. Virtual communities, crowdsourcing, social and entertainment sites can all provide important interaction, community and employment opportunities to large numbers of people. While Web 2.0 can provide enormous benefits, all of the new interaction paradigms are not immediately accessible.

The basic Web has become fairly accessible to people with disabilities but this was not always the case. Initially people using assistive technologies such as screen readers, screen magnifiers, and alternative input devices had difficulty interacting with the web. Requirements by governments have forced companies to address accessibility and to follow guidelines such as the W3C Web Content Accessibility Guidelines to make the Web accessible. However, the new Web 2.0 technologies are pushing the limits of assistive technologies.

Web 2.0 uses scripting and other advanced technologies to create visually appealing, highly interactive rich internet applications. Most of these applications are very visual and rely on mouse interactions to operate. Each Web 2.0 application wants to distinguish itself from others based on a compelling visual design, rich user interface and dynamic interaction. The incremental update of pages which can provide performance and real-time updates are not always accessible to people using assistive technologies. The assistive technology is not always able to interpret the user interaction model, nor is aware of the many updates occurring with a page or how to notify the user of the changes. Even for users able to visually interact with a site, the complicated interactions, and updates may be overwhelming or confusing. The use of additional semantics, adaptive interfaces and navigation options can make Web 2.0 more accessible.

3. Technologies to Enable an Accessible Web 2.0

Just as accessibility was not immediately in place when the Web first emerged, there is still work to be done to make Web 2.0 fully accessible. New specifications can add additional semantics into a Web page or application to enable assistive technologies to better represent the interfaces and interactions to the user. Extended accessibility application programming interfaces (APIs) will provide more comprehensive information to assistive technologies. The semantic Web will enable strategies to adapt the user interface to the specific needs of the user.

3.1 Accessible Rich Internet Applications

Accessible Rich Internet Applications (ARIA) is a specification being brought forward by the W3C Web Accessibility Initiative's (WAI) Protocols and Formats Working Group. The goal of ARIA is to add additional semantic data into HTML and XHTML to allow assistive technologies to better represent user interface components and dynamic interactions to the user. The specification also addresses providing input focus and full keyboard navigation within the components of an application. [7]

3.1.1 Providing Additional Semantics

Many of the elements in HTML have standard roles and properties which are known by the browsers and conveyed to assistive technologies via operating system accessibility APIs. These include link elements, form elements, lists, and headings. Web 2.0 requires more sophisticated components such as tree controls, tab panels, pop-up dialogs, rich text editing components, updated regions, on-line chat, and more. Most developers use generic elements such as <div> and <spans> with scripting to create these additional controls and interface elements. XmlHttpRequest allows the page and the components to be updated dynamically as data changes or for navigation over large data sets. An assistive technology has no semantic information about these created components and dynamic updates.

Desktop applications already implement many of these controls and the accessibility APIs have mechanisms to describe these components. The main idea behind ARIA is to add the necessary semantic data into the HTML and XHTML markup. The browser can then interpret this additional semantic data and provide it to the assistive technology via the accessibility API of the platform. Thus, a screen reader can identify a tree control as such. Each tree item is indicated as well as its hierarchy within the tree and its expanded or collapsed state.

The ARIA specification defines a standard set of roles and states that can be added into a component. Currently versions 1.5 and later of the open source Firefox browser implement the in-progress ARIA specification on the Windows platform.

3.1.2 Input Focus and Keyboard Navigation

Providing the semantic information about a component is the first step, in addition, the user must be able to navigate and interact with that control. Input focus and keyboard navigation is essential to allow people using assistive technologies to interact with a component as well as to support users with mobility issues. Assistive technologies need to track which elements on the page have focus and provide information about that component. The Document Object Model specification allows all elements to receive keyboard events, however, in current browsers only form and link elements receive input focus via the keyboard by default. The standard HTML mechanism is for keyboard users to navigate from focusable item to item via the tab key. With sophisticated components this tab key navigation is cumbersome and no longer practical. The ARIA specification defines the use of the tabindex key to indicate which elements may receive keyboard focus. The use of the tabindex attribute to enable focus was adapted from the implementation in Internet Explorer and support for the tabindex attribute has been implemented in Firefox as of version 1.5.

The tabindex attribute can be added to nearly any element. The value of the tabindex indicates how the element can receive focus. Elements with a tabindex of 0 are placed into the tab order of the page and can receive keyboard focus via the tab key. Form and link elements have intrinsic support for keyboard focus and do not require an explicit tabindex value of 0. Elements with a positive tabindex value are placed into the tab order before elements with an intrinsic tabindex or with a tabindex of 0. Elements with a tabindex value of negative one can receive focus programmatically. This programmatic focus allows Web developers to handle keyboard events and set focus to a specific element. This means that, via scripting, arrow key navigation can be implemented within components on the Web. User interface components on the Web can be implemented to work in the same manner as the desktop versions of these components.

Now a tree control, tab panel or other complex user interface component on the Web can be fully identified to a screen reader user. The component can receive initial focus via tab key navigation from component to component on the page. The user knows the type of control that has focus, its current state (expanded, checked, selected, etc.) and any additional properties such as grouping and hierarchy. The user navigates within the component via arrow keys in a similar manner to the desktop version of the component.

Figure 1 shows a tree control implemented within a sample Web application. The tree displays two top level nodes, Antarctic and Arctic. The Antarctic node has been expanded to reveal three child nodes; Penguins, Seals, and Whales. The Penguins node has been expanded to display five child elements, Adelie, Chinstrap, Emperor, Gentoo, and Rockhopper. The Adelie item has focus.

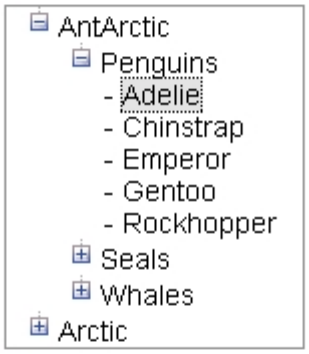

Figure 1 A Web Tree Control

Because this tree node has been implemented using ARIA techniques, the Firefox browser in Windows can communicate full information about this control to a screen reader.

The user navigated to the first node of the tree control, Antarctic, by pressing the tab key. When the Antarctic node received focus the Window-Eyes screen reader spoke, "Antarctic closed one of two tree view." Pressing the right arrow key expanded that node to reveal the children, "Antarctic expanded three items one of two tree view". Pressing down arrow put focus on the Penguin node, "Penguins closed one of three depth two". Pressing right arrow expanded that node, "Penguins expanded five items". Finally, one additional press of the down arrow focused the Adelie node, "Adelie one of five items depth three".

The keyboard operation makes it easy to navigate within this control and the ARIA implementation gives a screen reader user complete information about the status and hierarchy of the control. Compare this to a tree control implemented using links where the user must tab to each node in the tree, press enter to expand or collapse the item and receives no semantic information about the state and hierarchy of the control.

In this sample application, when enter is pressed with focus on a tree node, information pertaining to that node item is displayed on the right hand side of the screen using Ajax techniques. The next step is to inform the user of these dynamic updates to the page.

3.1.3 Dynamic Updates

One of the classic characterizations of Web 2.0 is dynamic updates to information. Ajax technologies allow pages to be incrementally updated. This may be in response to a user request, such as opening a menu, checking for new mail or updating an on-line instant chat. Web applications may also provide automatic updates such as providing updated stock quotes, sports scores, temperatures or other information. Via Web Services, a single Web page may be made up of information from several different sources. Information about these updates is not always available to assistive technologies. The ARIA specification addresses this via regions.

Different portions of the page can be identified with a role of region indicating that it is a perceivable unit which is given a title. Regions have properties which identify the type of region and how it is updated. It may be live or atomic. Live regions have additional semantics to identify the type of update and how the user should be notified. Users will not be notified of updates marked as polite until any current activity is completed. On the other side of the spectrum are updates marked as rude which are high priority and should interrupt any current activity. Support for regions is being added in to Firefox 3 and will require additional support from assistive technologies.

Through the use of scripting, semantic metadata, and input focus paradigms, the Accessible Rich Internet Application specification enables Web 2.0 applications and interfaces to become accessible to all users, including those requiring assistive technologies. It is imperative that the browser can communicate all of the necessary information about the Web components to the assistive technology.

3.2 Accessibility API's

Each operating system provides a set of Accessibility application programming interfaces to communicate information to assistive technologies. The assistive technologies need to know the details of components such as name, role, states, descriptions, etc, and to respond to events and changes to components in order to communicate with assistive technologies. Each operating system has its own accessibility API. There is Microsoft Active Accessibility (MSAA) and UI Automation for Windows, Linux Accessibility Toolkit (ATK) for Linux and MAC Accessibility API from Apple. There is even an accessibility API for Java. Web 2.0 is introducing new concepts which can not be handled by some of the current accessibility APIs, most notably MSAA.

The Microsoft Active Accessibility API was created many years ago and does not contain programming interfaces to represent some of the rich document editing and advanced features supported in Web 2.0 applications. IBM developed an extension to MSAA called IAccessible2 and, even though this is a Windows technology, has donated this open standard to The Linux Foundation [8]. IAccessble2 adds additional interfaces to MSAA to update it with equivalent functionality to the Java and Linux APIs. This will allow more uniform support for the Open Document Format (ODF) for describing electronic documents. Critical for Web 2.0 Ajax applications is that IAccessible2 provides the necessary interfaces to support ARIA live regions and dynamic editing. In addition to providing detailed information about the Web content and interactions, another mechanism is to adapt information and present it based on the abilities and preferences of the user.

3.3 Adaptation Strategies

The idea behind the Semantic Web is to provide data on the Web in a universal format that can be interpreted by software agents. This makes all data easily searchable and ubiquitous. A universal format also allows the data to be presented in different formats and modalities. [9]

Resource Descriptor Framework (RDF) is the language of the Semantic Web. It is a mechanism to represent resources in a manner that can be utilized by applications. RDF identifies items via uniform resource locators and describes them via properties and values. The RDF syntax is XML based and can be extended to represent any type of data [10]. RDF can enable device independence which will allow information to be presented on a variety of devices. This is important for the Mobile Web as well as for accessibility.

The Semantic Web Accessibility Platform (SWAP), created by UB Access uses RDF to create a knowledge based approach to accessibility [11]. SWAP adds accessibility notations to a Web page which are interpreted by a proxy server to adapt the data to a particular user.

Another project which is focused on adapting Web interfaces is the Fluid Project from the Adaptive Technology Resource Center at the University of Toronto [12]. It will use Web 2.0 technologies such as scripting and Ajax to customize the user interface based on the user's needs. The goal is to improve the user interfaces of academic software to address accessibility, usability and internationalization goals.

4. Developing an Accessible Web 2.0

New technologies which address accessibility of Web 2.0 are important but do not solve the problem until they are put into practice. Today, nearly anyone can create a presence on the web by using simple tools to create Web pages, blogs and wikis. Many interface providers will supply and install the necessary software so that little technical skill is required. Accessibility needs to be built into these tools. The first step is to enable toolkits with accessibility, then applications built using these toolkits will inherit accessibility.

4.1 Toolkits

There are several open source JavaScript and Ajax toolkits available to make creating Web 2.0 applications faster and easier. These toolkits make Web 2.0 development easier by abstracting browser differences and providing base functions for event handling, Ajax interactions, data binding, graphic effects as well as rich, customizable user interface components [13]. Examples of such toolkits include:

- Dojo http://www.dojotoolkit.org/
- Google Web Toolkit - http://code.google.com/webtoolkit/
- Open Rico- http://openrico.org/
- Prototype - http://www.prototypejs.org/
- TIBCO General Interface (available via an open source license) http://www.tibco.com/software/rich_internet_application/default.jsp
- Yahoo User Interface Library - http://developer.yahoo.com/yui/
- Zimbra Kabuki AJAX Toolkit (http://www.zimbra.com/community/kabuki_ajax_toolkit_download.html)

Others solutions such as DWR, Direct Web Roaming, (http://getahead.org/dwr) focus on accessing Java from JavaScript. Most of these toolkits understand that accessibility is an important issue and are working to integrate accessibility.

The core user interface component (widget) set of the Dojo Open Source toolkit is being updated for accessibility for the 1.0 release expected in the fall of 2007. Dojo will be one of the first toolkits to implement ARIA techniques to provide full keyboard and assistive technology support. General Accessibility information, strategy, Dojo accessibility resources, and ARIA implementation details have been documented by the author of this paper, in the Internationalization and Accessibility Chapter of the on-line Dojo Book [14]. Dojo plans to include documentation for creating and using each core widget, including accessibility considerations.

Although not an open source project, Bindows (http://www.bindows.com/) is a framework for Ajax applications developed with accessibility in mind. While it doesn't have some of the richly styled user interface components of other toolkits, it claims compliance for US. Government Section 508 requirements in Internet Explorer on the Windows platform.

4.2 Integrated Development Environments

In addition to toolkits, Integrated Development Environments (IDE) can make the creation of Web 2.0 applications easier and faster. Many even include rich user interface components with accessibility built in. IBM's Rational Web Developer is one example that includes a library of rich user interface elements which meet accessibility requirements. Microsoft's ASP.NET AJAX is a free framework for developing Web 2.0 applications that integrates with the Microsoft Visual Studio development environment. The Ajax Tooling Framework (http://www.eclipse.org/atf/) is an Eclipse plug-in to make development of Web 2.0 applications easier and in integrates with some of the open source toolkits mentioned previously. Tools for building accessible Web 2.0 applications are improving but testing these dynamic applications for accessibility requires advanced testing tools.

5. Testing an Accessible Web 2.0

Many testing tools exist to test the accessibility of the Web. These tools will evaluate the HTML of a Web site against a specific set of guidelines such as WCAG 1.0 or the US Government Section 508 requirements. They provide a report of the accessibility errors found which must be manually addressed by the developers. These tools are often criticized for providing too much information about some errors and can miss other errors. For example a tool can determine if alternative text is provided for an HTML img element by testing for the existence of an alt attribute. However, the tool can not ascertain if that alternative text is appropriate. Only the developer can know if an empty alt attribute was provided because an image is decorative only, or if it was just inserted because the development tool, trying to assist with accessibility, required at least some entry for the alt attribute when the img element was created.

Since most tools evaluate the resulting HTML pages, as more dynamic server side Web development technologies such as Active Server Pages (ASP), JavaServer Pages (JSP), Ruby on Rails, and PHP Hypertext Preprocessor are used, it is difficult for a tool to determine the exact source of the error. Add Web 2.0 dynamic updates into the mix and the testing strategy gets much more difficult. New tools are needed to address Web 2.0 applications.

5.1 Rule-based Accessibility Validation Environment (RAVEN)

Rule-base Accessibility Validation Environment (RAVEN) was introduced by IBM as a set of Eclipse plug-ins for verifying the accessibility of Java graphical user interface applications [15]. RAVEN relies on Aspect Oriented programming techniques to provide a non-invasive automatic to semi-automatic means to evaluate accessibility. It is based on an architecturally neutral validation engine which operates via an XML based set of rules. It integrates accessibility testing into the development environment but can also be used to test completed applications.

Due to the use of a pluggable architectural model, RAVEN can support other graphical frameworks in addition to Java based ones. RAVEN has been updated to support dynamic HTML which allows it to be used for testing Web applications. The integration of RAVEN into the Eclipse platform allows the evaluation of dynamic Web applications as they are being developed, and tested within the development environment. In addition, there are further plans to support the ARIA specification – enabling testing capabilities of Web 2.0 applications which utilize ARIA techniques [16].

5.2 Functional Web Accessibility Techniques and Tools

The University of Illinois has developed a set of Web Accessibility Best Practices as well as a Functional Web Accessibility Evaluator (FAE) Tool to test a Web sites use of the Best Practices [17]. The Accessibility Best Practices are based on a set of five principles, Navigation and Orientation, Text Equivalents, Scripting and Automation, Styling, and Standards Coding techniques are provided to implement these principles. Unlike traditional tools which search for specific tags and attributes, such as img tags with no alt attribute, the FAE tool evaluates based on the coding techniques recommended in the Best Practices - essentially applying the Best Practices coding examples as rules for the evaluator.

The University has also developed a Mozilla/Firefox Accessibility Extension which provides visual feedback about the accessibility features of a Web resource and can also be used with the FAE. Using the Accessibility Extension a developer, tester, or user can turn on and off different features used on the Web site such as images or Cascading Style Sheets (CSS). In addition, information about different structural features such as headers, alternative text, and labels can be visually highlighted or displayed in dialog boxes. This is accomplished by querying the Document Object Model (DOM), allowing for the dynamic testing of Web applications. In addition, the Mozilla/Firefox Accessibility Extension queries and displays information about the correct implementation and use of ARIA techniques. Having development and testing tools for ARIA is one of the key steps in encouraging the adoption of this specification and enabling an accessible Web 2.0.

6. SUMMARY

The dynamic nature of Web 2.0 is creating challenges for accessibility. Users of assistive technologies may be unaware of the behavior and operation of dynamically created user interface controls so prevalent in Web 2.0. Ajax based applications which dynamically update portions of the page create additional difficulties. In order to make Web 2.0 accessible to all users, more semantic information and behaviors need to be embedded into Web applications and provided to assistive technologies. The ARIA specification provides a mechanism to add the additional semantics and notifications. Future use of RDF and adaptation strategies will further enable accessibility, unified searches and usability. Enhancements to accessibility APIs, and their adoption by browsers and assistive technologies will allow these new Web 2.0 paradigms to be exposed to all users. The integration of accessibility technologies into development environments, toolkits and testing tools can make creating accessible Web 2.0 applications the norm.

7. REFERENCES

[1] Chisholm, W., Vanderheiden, G. and Jacobs, I (1999) Web Content Accessibility Guidelines 1.0, http://www.w3.org/TR/WCAG10/

[2] O'Reilly, Tim, (2005) http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html

[3] Security Space Technology Penetration reports, http://www.securityspace.com/s_survey/data/man.200103/techpen.html and http://www.securityspace.com/s_survey/data/man.200703/techpen.html.

[4] Regan, Bob, (2005) Macromedia White Paper, *Best Practices for Accessible Flash Design*, http://www.adobe.com/resources/accessibility/best_practices/best_practices_acc_flash.pdf

[5] Howe, Jeff, The Rise of Crowdsourcing. Wired Magazine, Issue 14.06, June 2006.

[6] Brewer, Judy, (2005) How People with Disabilities Use the Web, http://www.w3.org/WAI/EO/Drafts/PWD-Use-Web/20050505

[7] Gunderson, J, Schwerdtfeger, R. (2006) Roadmap for Accessible Rich Internet Applications (WAI-ARIA Roadmap), http://www.w3.org/TR/aria-roadmap/

[8] The Linux Foundation, The Free Standards Group to Standardize New Accessibility Interfaces, Press Release, December 14, 2006, http://www.linux-foundation.org/wordpress/?p=276

[9] Berners-Lee, T. Hendler, J., Lassila, O. The Semantic Web, Scientific American, May 2001.

[10] Manola, F.,Miller E. (2004) RDF Primer, http://www.w3.org/TR/rdf-primer/

[11] Seeman, L. The Semantic Web, Web Accesibility, and Device Independence. ACM SIGCAPH Newsletter, No. 76, June 2003.

[12] Smith, B. University of Toronto leads project for adaptive Web applications, Canadian Technology News, April 12, 2007 http://www.itbusiness.ca/it/client/en/home/News.asp?id=43009 and http://www.fluidproject.org/

[13] Wayne, P. Surveying open-source AJAX toolkits, InfoWorld, July 2006, http://www.infoworld.com/article/06/07/31/31FEajax_1.html

[14] The Dojo Book, Chapter 8: Intenationalization and Accessibility, http://www.dojotoolkit.org/docs/book/part8

[15] Feigenbaum, B., Squillace, M., (2006) Accessibility Validation with RAVEN. In *Proceedings of the 2006 international workshop on Software Quality,* ACM Press (pp 27-32).

[16] Squillace, M. IBM Rule-based Accessibility Validation Environment, presentation at California State University at Northridge (CSUN) Technology & Persons with Disabilities Conference, March, 2007, Los Angeles, CA

[17] Gunderson. J., Rangin, H. D., Hoyt, N. (2006), Functional Web Accessibility Techniques and Tools from the University of Illinois. In *Proceedings of the 8th International ACM SIGACCESS Conference on Computers and Accessibility ASSETS 06,* ACM Press (pp 269-270).