

Ajax Live Regions: ReefChat Using the Fire Vox Screen Reader as a Case Example

Peter Thiessen
Adaptive Technology Resource Centre
University of Toronto
130 St. George St., Toronto, Ontario
peter.thiessen@utoronto.ca

Charles Chen
Empirical Software Engineering Lab
The University of Texas at Austin
713-557-7289
clc@clcworld.net

ABSTRACT

Web 2.0 enabled by the Ajax architecture has given rise to a new level of user interactivity through web browsers. Many new and extremely popular Web applications have been introduced such as Google Maps, Google Docs, Flickr, and so on. Ajax Toolkits such as Dojo allow web developers to build Web 2.0 applications quickly and with little effort. Unfortunately, the accessibility support in most toolkits and Ajax applications overall is lacking. WAI-ARIA markup for live regions presents a solution to making these applications accessible.

To address this problem we developed an Accessible Ajax chat application called ReefChat and the Fire Vox screen reader. Features include, chat message notification through live regions to notify the AT. As well as keying up and down messages to navigate through chat messages, and keying left and right to filter messages from specific users. In this paper after briefly discussing the problem of Web 2.0, we describe our accessible chat application and screen reader.

Categories and Subject Descriptors

H.1.2 [Models and Principles]: User/ Machine Systems—*human factors, human information processing*; **K.4.2 [Computers and Society]:** Social Issues—*assistive technologies for persons with disabilities*

General Terms

Human Factors, Design, User Agents

Keywords

Accessibility, Web 2.0, Ajax, ARIA, Live Regions, User Agents

1. INTRODUCTION

Traditionally, Assistive Technologies (AT) have treated information on a web page as content that can be linearized. Ajax web applications break this assumption; new content can appear in arbitrary locations and user interactions with the page are far more complex. Since these Ajax web applications behave more like desktop applications than web pages, solutions for making

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

W4A2007 - Challenge, May 07–08, 2007, Banff, Canada. Co-Located with the 16th International World Wide Web Conference.

Copyright 2007 ACM 1-59593-590-8/06/0010 ...\$5.00.

desktop applications accessible can be applied to these Ajax web applications. One of the most important aspects of making desktop applications accessible is to inform users of important events that are occurring on parts of the screen, even if those parts are not focused. For example, in a chat application, the user's focus is on the input blank, but it is essential to inform the user of what the other chatters have typed. On the other hand, it is important not to overwhelm the user with a flood of information, especially if that information is trivial.

In traditional desktop applications, there are a set of known widgets such as buttons, trees, data cells, etc. These widgets behave in a predictable manner; thus, an AT simply needs to know how to support the events of a type of widget in order to provide reasonable support for any instance of that type of widget. However, Ajax applications do not share this uniformity. Many Ajax applications use custom widgets created out of span and div elements, mixed with input elements and graphics, and laid out by CSS. Sometimes, Ajax applications will even use custom widgets for standard HTML widgets such as a button because the application developer wished to change the behavior and/or appearance of that widget to fit the particular application better. As a result, while AT can pick up DOM mutation events, it is very difficult, if not impossible, for an AT to understand what that event represents in the context of an Ajax application.

The solution to the DOM accessibility problem is to markup the live regions, the regions on the page which can be changed by Ajax. Markup for live regions is part of the Web Accessibility Initiative - Accessible Rich Internet Applications guideline (WAI-ARIA) [5]. Live regions [3] are only one part of ARIA, other parts enable desktop-style Javascript widgets, specify typing restrictions on data, or mark regions of a page with landmarks (such as the main content). Future versions of ARIA are expected to allow accessible diagrams as well as author-defined roles, properties and relations.

2. ACCESSIBLE CHAT

We developed an Accessible Ajax chat application called ReefChat [4]. The goal of this chat is to demonstrate that a Rich Internet Application (RIA) could be both accessible and aesthetically appealing. Reef Chat has currently reached the first phase of development and has the basic functionality of a typical accessible chat application. The chat is targeted to work specifically with screen readers and follows the WAI-ARIA [5] guidelines and more specifically, the ARIA Live Region markup to expose chat events to the DOM. The future roadmap of ReefChat includes converting it into a Dojo [1] widget.

Charles Chen is the creator of the CLC-4-TTS Suite [2]. One of the applications in this suite is Fire Vox, an open-source, freely-available, talking browser extension for the Firefox web browser.

It is cross-platform compatible and can run on Windows, Macintosh, and Linux. The only part of ARIA that is currently supported by Fire Vox is the live region markup. Window-Eyes and JAWS support the widget-related markup, but not the live region markup. This is the limitation of ARIA support in current AT products.

2.1 CHAT FEATURES

ReefChat is still in its infancy and has only the very basic chat features. An important feature is the ability to have new chat messages exposed to an AT using live regions. The markup for a typical chat message is displayed Figure 1.

```
<div class="chatMessage" aria:role="chatMessage">
<div aria:state="timestamp" class="timestamp">23:24:25</div>
<div aria:role="username" class="username">peter</div>
<div class="content">Live regions make the chat go round!</div>
</div>
```

Figure 1. Chat Message Markup

The ARIA Role chatMessage inherits from the Group Role but does not currently exist as an ARIA Role (we are hoping it will be included in the future ARIA guidelines). Chat messages are placed inside a Live Region which allows Fire Vox and any Live Region compliant AT to speak the chat messages, the markup is displayed in Figure 2.

```
<div id="transcript" aria:role="log" aaa:live="polite"
relevant="additions" atomic="false">
```

Figure 2. Chat Transcript Live Region Markup

The live region settings allow chat message to be spoken in sequence, and individually as they are added.

The User List uses the same Live Region Markup to inform ATs of new users but also when users log off, so the markup would be relevant="additions removals".

Fire Vox has many useful features, one being a summary interface of page headings. Hitting Ctrl + Shift + H displays this interface and in the case of ReefChat, would display shortcuts to the headings ReefChat: Messages, Compose Message, User Listing, and Options. The interface allows the user to quickly navigate between the different chat User Interface (UI) elements.

The chat displays and exposes to the AT each message as it is received. Navigating to the Chat Messages section, a user can use the up and down keys to navigate through the chat message log. If the user decides to halt the screen reader, s/he can key Ctrl + Shift + C at any time, and continue reading all Web page elements with Ctrl + Shift + A.

A very useful innovation is the ability to filter chat messages based on username. Suppose an active chat is underway or a spammer enters the room, a user can filter only the messages from a specific user to be spoken. This is done by keying the left and right keys to navigate through messages from a specific user.

The chat in figure 3 for example, a user chose to only display chat messages from erin.

The screenshot shows the ReefChat interface. On the left, there is a 'ReefChat: Chat Messages' section with a list of messages. The messages are:

- 23:33:42 System: peter has entered the chat room
- 23:33:53 System: charles has entered the chat room
- 23:34:15 System: aaron has entered the chat room
- 23:34:25 System: simon has entered the chat room
- 23:34:48 System: erin has entered the chat room
- 23:35:00 peter Greetings all!
- 23:35:10 charles Hi, hitting ctrl shift h displays the headings and allows you to quickly the interface from Chat Messages to Compose Messages and so on
- 23:35:20 aaron Wow I can key up and down the messages manual and have them spoken to me
- 23:35:32 simon Neat, I can use the left and right keys to filter messages based on a user name. Not that I'd want to filter anyone's messages here ;)
- 23:35:53 erin I can hear users entering and leaving the chat also in the User List area
- 23:36:10 charles I need to leave to catch my flight for the W4A conference - ttyl

 On the right, there is a 'User List' section with the following users listed:

- peter
- charles
- aaron
- simon
- erin

 Below the chat messages, there is a 'Compose Message' section with a text input field, a 'Send' button, and an 'Options' section with a 'logout' link.

Figure 3. ReefChat - Chat Example

Doing so allows a user to quickly hear all messages from Erin.

3. CONCLUSION

The problem of accessibility for Web 2.0 Internet applications can be solved by complying with the WAI-ARIA guidelines. ReefChat demonstrated how Live Regions can be used to expose DOM events to an AT such as Fire Vox. The features in the Fire Vox screen reader were also demonstrated and allowed a user to efficiently access the chat information.

4. ACKNOWLEDGMENTS

Our thanks to the Mozilla Foundation for funding our projects and research.

5. REFERENCES

- [1] Dojo. "Dojo the Javascript Toolkit". 15 January 2006 <<http://dojotoolkit.org>>
- [2] Chen, Charles. Fire Vox. 11 January 2007 <<http://firevox.clcworld.net>>
- [3] Mozilla Developer Center. "AJAX:WAI ARIA Live Regions". 23 January 2007 <http://developer.mozilla.org/en/docs/AJAX:WAI_ARIA_Live_Regions>
- [4] Thiessen, Peter. ReefChat. 20 March 2007. <<http://reefchat.overscore.com>>
- [5] World Wide Web Consortium (W3C). Roadmap for Accessible Rich Internet Applications (WAI-ARIA Roadmap) 20 December 2006. 17 January 2007 <<http://www.w3.org/TR/aria-roadmap>>