Tailored presentation of dynamic Web content for audio browsers.

Andy Brown[a,*], Caroline Jay[a], Simon Harper[a]

[a]*School of Computer Science, University of Manchester, Kilburn Building, Oxford Road, Manchester. M13 9PL. UK*

Abstract

Understanding the content of a Web page and navigating within and between pages are crucial tasks for any Web user. To those who are accessing pages through non-visual means, such as screen readers, the challenges offered by these tasks are not easily overcome, even when pages are unchanging documents. The advent of 'Web 2.0' and Web applications, however, means that documents often are not static, but update, either automatically or due to user interaction. This development poses a difficult question for screen reader designers: how should users be notified of page changes? In this article we introduce rules for presenting such updates, derived from studies of how sighted users interact with them. An implementation of the rules has been evaluated, showing that users who were blind or visually impaired found updates easier to deal with than the relatively quiet way in which current screen readers often present them.

Keywords: Web 2.0, AJAX, Visual Disability, Eye-Tracking

1. Introduction

Screen readers are a common means for users with visual impairments to access electronic information, including Web content. A screen reader is a piece of software that uses synthetic speech (and non-speech sounds) to render the information in an audio form (Alliance for Technology Access, 2000; Burks et al., 2006; Raman, 2008; Chen, 2006). It also enables the user to move around the information at different levels of granularity, for example, between letters, words, lines, paragraphs, or sections, and to jump between different types of object (e.g., heading or hyperlink). Although complex pages remain challenging to understand (particularly so if the author has not made good use of semantic markup), users are normally able to get the gist of a page and find the information they require.

*Corresponding author

Email addresses: andrew.brown@cs.manchester.ac.uk (Andy Brown), caroline.jay@manchester.ac.uk (Caroline Jay), simon.harper@manchester.ac.uk (Simon Harper)

The Web is changing, however ('Web 2.0' (Oreilly, 2007)), and pages are becoming more interactive, sometimes resembling applications more than documents. These changes are achieved by techniques such as AJAX (Asynchronous JavaScript and XML), that allow client–server communication without needing a full page refresh, so that regions of a page may update independently (Mahemoff, 2006). This is proving problematic for screen reader users (Zajicek, 2007; Thiessen and Chen, 2007; Brown and Jay, 2008), as their technology lags behind Web development. It also poses difficult questions for screen reader developers, who need to change their interaction model to cope with detecting and presenting updates. The current situation is that screen readers generally do not present updates to the user, and when they do it is done in a simple way, neither accounting for the content of the update nor the user's activity. The problem is such that many of these users have not *knowingly* encountered many common types of updating content. This lack of awareness is probably not due to them not having visited pages with dynamic content, but to them having been unable to identify the content as such. A typical scenario might be as follows. A user encounters what seems to be a link to content they would like, and clicks it. The link, however, is not a traditional link to another page but a control to update another part of the page, and since the screen reader does not inform them of the update, it appears to the user as if nothing has happened. The user continues to browse, needing to access the information in a different way, and unaware of having encountered dynamic content.

Currently, those efforts to improve accessibility of dynamic updates have focused on the page creation process (Keith, 2006; Thiessen and Chen, 2009), the most significant contribution coming from the World Wide Web Consortium (W3C) Web Accessibility Initiative (WAI). They are coordinating the development of markup that allows developers to annotate their 'rich content' (i.e., controls and updating regions) with information that screen readers (and other assistive technologies) can use to aid presentation. This is known as Accessible Rich Internet Applications, or WAI-ARIA (Gibson, 2007). The ARIA markup can be broadly split into two areas: that which makes the controls keyboard accessible and their roles clear; and that which deals with the areas that update ('live regions'). The tags associated with live regions allow assistive technologies to be given information about an update, including how important the update is (its *politeness*: polite, assertive, or off), how much of the page needs to be re-presented to the user after the update, and how the DOM is affected by the change. Support for ARIA is now included in the major browsers (although it is often limited) and assistive technologies (at least partial support in recent versions of Orca, NVDA, Window Eyes and Jaws).

Screen reader developers in both commercial and academic environments are also tackling the difficulties arising from updating pages, not only by supporting ARIA, but also with better general handling of updates. Of particular note is the Hearsay Dynamo system (Borodin et al., 2008), which treats inter and intra-page changes in the same way, allowing users to maintain focus when moving between similar pages and when part of a single page updates. In this system, users are notified of an update using a short sound clip, and are given commands

that allow them to navigate to the new content. Users are not notified about updates involving removal of content. Evaluation of this system showed that it improved access to the updates.

While access might be becoming more achievable in the technical sense, however, it is still the case that visually impaired users are not receiving the benefits that *efficient* access to updating pages could bring. The work described here can be seen as complementing the ARIA approach. We believe that, while the information the ARIA attributes and properties provide can help, it cannot be used in a naïve manner. Thiessen and Chen (2007) found that ARIA had limited ability to make frequently changing pages (their example was a chat room) accessible, although further developments in ARIA and its application are improving this (Thiessen and Chen, 2009; Thiessen and Russell, 2009). Understanding how to use ARIA tags, both from the point of view of the developer (which settings to choose) and from that of the user-agent and assistive technologies (how to use the tags to help determine exactly when and how to present an update), is difficult, and we believe that this process must be informed by an empirical understanding of how sighted users interact with updates.

1.1. Contribution

Our motivation is to enable users to handle dynamic content in the most efficient way. In this context, this means supporting the user in: becoming aware of updates; determining their usefulness; accessing their content (if desired); and resuming or modifying the task at hand. This must all be done with minimum disruption to the primary task.

This paper describes a novel system for tailoring the presentation of Web page updates — presenting each update according to its characteristics rather than applying the same rule to all. The approach taken in the work described here is to base an accessibility solution on a solid understanding of both the problems faced by screen reader users, and the benefits that sighted users gain from the visual presentation. A brief overview of the differences between exploring information in audio and visual media highlights the main problems for screen reader users as being: difficulty gaining an overview, glancing and navigating; and a lack of peripheral awareness (§ 2). An analysis of types of dynamic content results in a classification according to how the update was initiated and how it affected the page (§ 3). This demonstrates that there is a wide range of updates, which are unlikely to be equally important to the user.

An eye-tracking study, exploring how sighted users interact with such updates, confirmed that updates are not all equal, with automatic updates receiving little attention and user-initiated ones almost always attended (§ 4). This study not only gave a quantitative model of attention, based on the class of update, but also a qualitative understanding of how certain updates benefited users. From this understanding comes the main contribution of this work: a set of mappings that can be used as general rules about how to present updates, as well as more specific interfaces for pop-up calendars and auto-suggest lists (§ 4.1 – 4.10).

An implementation of these rules was developed for their evaluation (§ 5). The system identified changes to a Web page, grouped them into meaningful updates, and classified them according to the taxonomy. This allowed the presentation rules to be applied, so that each update had its presentation tailored according to its class. This system was subject to an evaluation by users with visual impairments (§ 6). It was compared to a system that resembled current popular screen readers. The system using mappings to tailor presentation was found to be easier to use and was preferred by all participants. These rules can be used to inform the design of screen-readers, suggesting an effective way to deal with updates. The results and approach could also help Web developers to apply appropriate WAI-ARIA markup to their code.

2. Audio Presentation: The Problems

In broad terms, the goal of this research is to enable users to interact with updating content in the most efficient way possible — to minimise the disruption caused by unhelpful updates, and maximise ease of access to necessary information. People who are browsing Web pages using a screen reader have several disadvantages compared to those exploring visually. The ultimate causes are that the audio information flow is essentially linear, and that it does not provide users with an external memory (Scaife and Rogers, 1996).

The visual representation provides a two-dimensional indexed external representation that is (relatively) permanent; thus briefly looking at an area away from the current locus of attention only requires the memory to hold the positions of the current position and the position of the desired information. With Web pages, there are typically many visual landmarks to ease this, but for the audio browser landmarks are much less salient and are only present in a one-dimensional stream. Furthermore, movement around this stream is slower. These differences combine to make jumping around a page either difficult or time-consuming, for the following reasons:

- Gaining an overview is difficult. The inability to move the attention rapidly around a scene (i.e., the Web page) means that it is difficult to get a quick, high-level, understanding of what information is available.
- Glancing is difficult. For the same reasons that the audio representation cannot effectively act as an external memory, it is difficult to glance at information away from the current locus of attention.
- Navigating around a scene is relatively slow and difficult. Although screen readers allow users to jump between different HTML elements (e.g., headings), moving to a precise point requires the user to keep the structure of the document in mind, move to the appropriate section, then navigate by line or sentence and word to the point of interest.
- There is no peripheral awareness. Although the visual perception system has a focus, it can also detect stimuli outside of this relatively narrow field.

This is particularly the case for areas with certain characteristics, such as colour or movement — these are said to have high visual salience (Carmi and Itti, 2006; Parkhursta et al., 2002). With typical audio interfaces this is not the case - the user listens to the word currently being spoken, and is not aware in any way of the current state of the rest of the document. It is possible for the designer of an audio interface to present information using background sounds (typically, but not necessarily, non-speech) to give users this type of information, but there are crucial differences between this and peripheral vision: the former need to be designed and learnt, and give less information (e.g., it is difficult to convey attributes such as location) than the latter.

It is clear that each of these problems is relevant when dealing with updating Web content, and that some of them are particularly acute. The lack of peripheral awareness means that users will not notice updates unless they are either made explicit (notification) or the new information is discovered (and noticed to be different) serendipitously. The discovery method is not conducive to efficient browsing: discovery may not happen, and if it does, the discovery of the effect (and hence that an update has occurred at all) is removed from its cause. Confusion may occur when an action appears to have no effect, and disorientation may occur when content has unexpectedly changed. Equally, the higher costs associated with resuming a task mean that there are problems associated with notification. Awareness of updates is crucial to efficient browsing.

The difficulties associated with glancing make the next step in the process difficult. Once an update has been observed, sighted users are able to assess the content relatively rapidly: a quick glance will take them to the new content, followed by an overview to answer the question "is this chunk useful to me?". As we have seen, however, both glances and overviews are problematic for screen reader users.

Consideration of these theoretical factors gives us the fundamental design goals: extra support is necessary for users with visual impairments, both in becoming aware of updates and in assessing their usefulness or interest.

3. Dynamic Web Pages: A Classification

Web 2.0 is a loosely-defined term, that differentiates the Web of the late 2000's from how it was in the 1990's. Two of the more significant features of Web 2.0 are user-generated content and Web applications. Both of these inevitably result in more interactive pages than was found in the original model (a collection of linked, but unchanging, documents); modern Web content can often be edited, combined or created by the user. One of the key technologies behind these changes is AJAX — asynchronous JavaScript and XML (extensible markup language) — a combination of older technologies that allows a Web page to communicate with a server, and change its content, without visiting a new page or refreshing the old one.

In order to understand the range of updates, we have developed a two-axis classification. This is based on what initiates the update and what effect it has on the page. A brief explanation of this classification is followed by examples.

The first axis of classification categorises updates according to how they were initiated (the cause of the update). This has two broad classes, with one further divided into two. First, updates may occur automatically, independent from any user activity (except, of course, loading the page). A typical automatic update might replace content according to a timer (e.g., a 'ticker' showing news headlines). Note that some content, particularly sports commentary, may update on a regular basis, but with not all updates resulting in a meaningful change. The second class is for updates which are triggered by some user activity, typically mouse or keyboard events. This can be sub-divided depending on the users task: some updates are explicit requests, i.e., the user is asking for more information (following a link, clicking a button, etc.); other updates are a side-effect of the action. Examples of the latter include auto-suggest lists (a list of suggestions given to the user while entering data into an input field) and form validation error messages; in both cases the update is triggered by the user (typing, or pressing the submit button), but getting the information provided by the update was not the primary purpose of that action. We designate these two categories user-requested and user-initiated.

The second axis is based upon the effect the update has on the host page, and has four classes. Information may be added, removed, replaced, or rearranged. The last category is reserved for 'semantic' rearrangements, where the page model changes, not just the layout (addition updates may require other page content to move to accommodate, but the relationships between all the original content remain constant).

To give examples of the updates in this classification, we present the dynamic content from the Web pages used in the evaluation. These pages simulated a holiday booking Web site, where participants could view special offers, and search for holidays. Table 1 summarises the taxonomy and shows how each type of content is classified. The following sections describe the content, and how users interact with it, in more detail. This is done from the perspective of a sighted user — the audio interaction will depend on the screen reader (the interaction techniques proposed in this paper will be described in section§ 5). The different types of dynamic content are described in the order in which they were encountered during the evaluation.

3.1. Ticker

The ticker provided the latest news from the HCW Travel Company. It displayed a single sentence of news at a time, and updated every five seconds (see Figure 1), looping through three headlines.

3.2. Tabs

A box in the centre of the HCW Travel home page contained the main content. The content could be changed by clicking one of three tabs along the

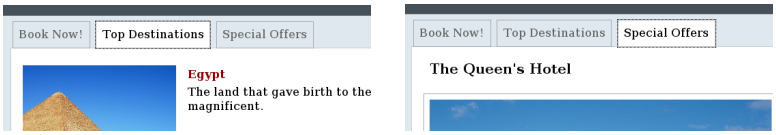
Update class			
Effect	**Cause**		
	Requested	Initiated	Automatic
Insertion	Expansion button (§3.4)	Pop-up calendar (§3.6); Form completion error message (§3.7)	Sports commentary — new comment.
Removal	Contraction button (§3.4)	Form corrected (§3.7)	
Replacement	Slideshow (§3.3), Tabs (§3.2)	Form input suggestions list (§3.5)	Ticker (§3.1)
Rearrangement	Table re-ordering (§3.8)		(Live 'most popular' list)

Table 1: Examples of dynamic content for each category in the classification. Updates are classified along two axes, according to the cause (how the update was initiated) and the effect (how the update changes the page).

(a) The news ticker. (b) The ticker 5 seconds later

Figure 1: The regularly updating ticker of latest news.

top of the box (see Figure 2): 'Book Now' revealed a link to the booking page; 'Top Destinations' revealed a list of recommended destinations; 'Special Offers' revealed a slideshow of special offers. The 'Top Destinations' tab was selected initially.



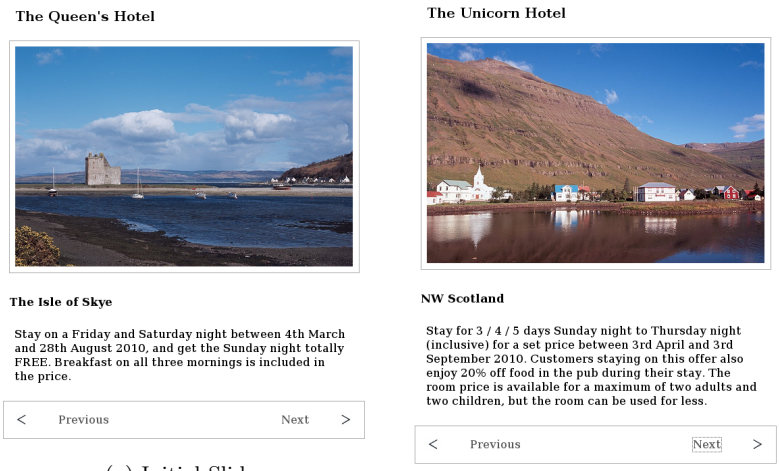
(a) Top Destinations tab (b) Special Offers tab

Figure 2: The effect of clicking the the 'Special Offers' tab heading.

3.3. Slideshow

The 'Special Offers' tab contained a series of holidays stored in a slideshow (this type of presentation is sometimes called a carousel). Each slide contained the name of the hotel, a photograph of the hotel, the name of the location and a description of the offer. Below the slide were two controls that could be used

to navigate between four slides in a loop. 'Previous' moved to the preceding offer and 'Next' moved to the next offer (see Figure 3).



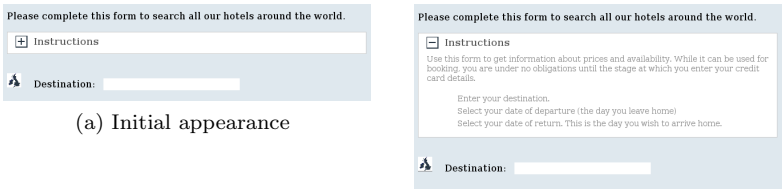
(a) Initial Slide

(b) Second Slide

Figure 3: The effect of clicking the control 'Next' on the slide show. The rest of the page remains unchanged.

3.4. Expansion button

A box contained a control entitled 'Instructions'. When this was selected, the box expanded to reveal a bulleted list of instructions describing how to use the form (see Figure 4). Selecting the control again removed the instructions — this type of update would be classified as requested-removal.



(a) Initial appearance

(b) Expanded appearance

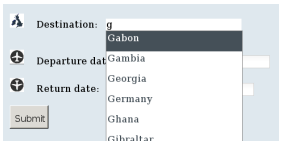
Figure 4: The effect of selecting the expansion button.

3.5. Auto-suggest list

When the user typed in the 'Destination' text field, a list of suggestions for holiday locations appeared under the text field. The suggestions were destinations that matched the characters that had been entered so far. Selecting a

suggestion entered it into the text field. Each character typed caused the suggestions list to be refreshed and potentially replaced with a new set of suggestions. The first appearance of the list is therefore initiated-addition. subsequent changes are initiated-replacement (see Figure 5).

(a) Initial appearance



(b) A suggestions list appears after typing.

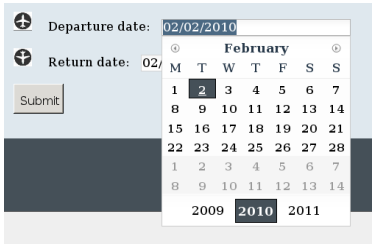
(c) The list changes according to the input.

Figure 5: The appearance of a suggestions list on typing in the destination field.

3.6. Calendar

When the focus entered one of the date fields a pop-up calendar appeared. Initially the current date was displayed in the date field, and this was visually highlighted in the calendar (see Figure 6). Two controls could be used to move to the preceding or next month, as in the slideshow. Selecting a date with the mouse entered it into the date field. When a departure date had been entered, the return date was automatically set to this too, providing it had been correctly formatted, otherwise it remained set to the current date.

(a) Date entry field without focus

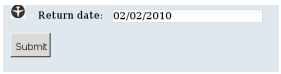


(b) Field focused

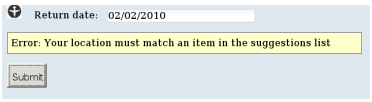
Figure 6: The appearance of the calendar on focusing the date entry field.

3.7. Error messages

To perform a search for a holiday, the user had to press the 'Submit' button at the bottom of the form. If the destination or date had been entered incorrectly (if the destination did not match an item from the list exactly, or the date was not formatted correctly), a box containing an error message appeared just above the 'Submit' button (see Figure 7).



(a) The lower section of the form



(b) The dynamic error message

Figure 7: The effect of submitting the form with incomplete information.

3.8. Table re-ordering

The results of the holiday search were stored in a table. Each row contained a holiday, and details of the hotel, location, room type, package, quality (star rating), customer rating and price were stored in separate columns. Pressing enter when the focus was on one of the column headings reordered the holidays from lowest to highest for that particular category (see Figure 8).

(a) Table on first appearance

(b) Reordered by customer rating

Figure 8: The effect of clicking the the 'Customer rating' column heading.

4. Effective Audio Presentation — Mapping the Visual Experience

Sighted user studies (Jay and Brown, 2008) have given us an understanding of how these users interact with the different types of dynamic content introduced above. Two studies were performed. In the first, 20 participants viewed either a static or dynamic version of each of three websites. These were identical except that some regions of the dynamic pages changed automatically whilst being viewed. In the second study 30 participants performed a series of directed and browse-based tasks on some live websites that were chosen to contain a significant amount of dynamic content. In both studies participants eye-movements were tracked, and the data used to identify which regions of the page they fixated, and when. These studies resulted in a quantitative model of what types of content users attend (summarised in Figure 9, the output from the SPSS Chi-Squared Automatic Interaction Detector tree classification procedure), as well as a more qualitative understanding of how users use, and benefit from, certain types of content. The findings of these studies can be summarised as follows:

- Automatically changing content was not frequently viewed.
- Requested content was viewed immediately. This was facilitated by visual cues and the salience of the change.
- Updates that assisted with input (e.g., calendars and suggestions lists) were used.

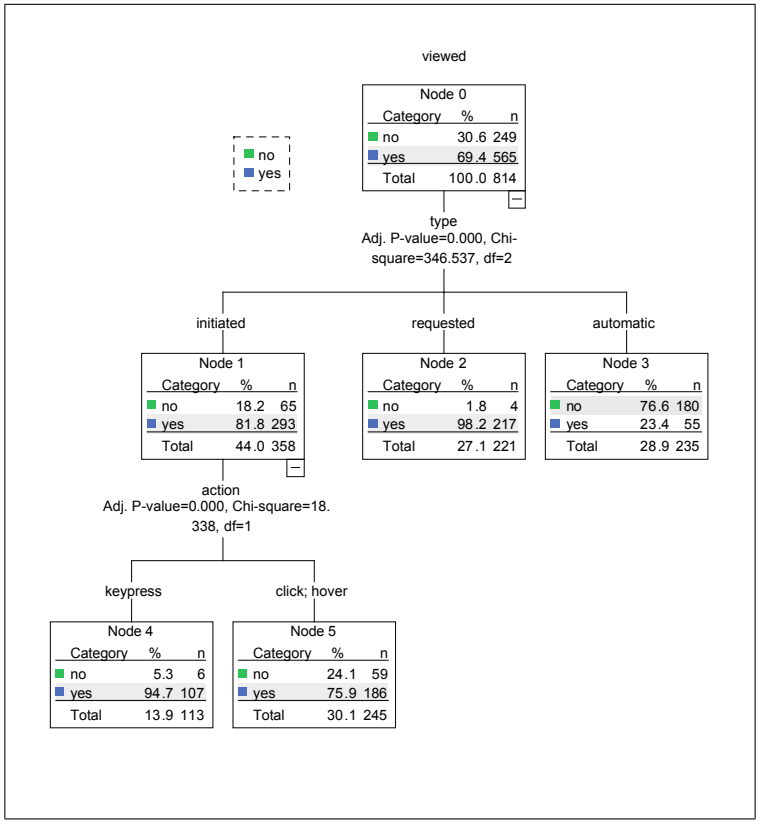


Figure 9: Sighted user studies model. This tree is the output of the SPSS Chi-Squared Automatic Interaction Detector tree classification procedure.

- The top of new content, but not necessarily the rest, was viewed.
  - Only the top three items in suggestions lists were regularly viewed.
  - After table rearrangements, people looked immediately at the top left cell.
- The tabular arrangement of pop-up calendars made relationships between dates explicit, while mouse selection helped avoid typing errors, and removed the need to deduce the date format.

Section 3 demonstrated that dynamic updates come in many forms. While sighted users are generally able both to notice that an update has occurred and to assess it for usefulness, an understanding of the additional problems faced by screen reader users demonstrates that for these users these are difficult tasks and need to be supported. It also becomes clear that supporting the user in deciding on the usefulness of the information in an update will demand a delicate

balance. On the one hand, a decision can only be made if the user is aware of an update, so some means of notification will be necessary. On the other hand, the extra demands caused by the lack of external memory mean that it is more difficult to return to the main task following any disruption.

These theoretical considerations suggest that it is not *desirable* to present all updates to users. Such a 'tailored presentation' approach is supported by the eye-tracking study, which suggests that it is also not *appropriate* to present all updates: sighted users routinely ignore changes to the page they are viewing. The question now becomes: which updates should be presented, and how?

In the perfect world, only those updates that the user decides are interesting or useful will be presented, while the rest will be quietly ignored. Unfortunately, automatic systems, such as screen readers, have very limited knowledge about the user's task, and even less about what he or she will want to read. In this case, therefore, an alternative approach is for the screen reader to monitor the user's actions and the content of updates, and apply a set of rules to determine when and how to notify the user of each. This is the approach taken here. The remainder of this section describes the rules that were derived from the findings of the eye-tracking study.

It is already apparent (Borodin et al., 2008) that users benefit from being notified about updates, but how to present them is less clear. Borodin et al. presented all updates containing new content equally, giving a brief non-speech sound, then allowed the users to navigate to the content if they wished. They noted that users were not entirely satisfied with this, but justified their decision on the need to maintain the user's orientation:

> "Several participants had suggestions on how the system could better convey the updated content to them. For instance, some wanted the system to automatically jump to the portions of the page that had updated. Although we developed the capability in HD to automatically jump to updated content, we did not evaluate it because we believed that overriding user intent would be too disorienting in general."

Our eye-tracking results, however, suggest that orientation might be maintained if automatic jumps are applied to certain, but not all, updates. In the study, sighted users rarely fixated automatic updates, but normally did fixate the first part of new content where the update was initiated by their actions. We propose, therefore to tailor the non-visual presentation in a way that matches this behaviour.

Presentation is dependant upon the class of update, both in terms of its effect on the page and its initiation type. The basic principle is derived from the model (Figure 9) and can be summed up as: provide a simple notification for automatic updates, move the focus and start speaking the requested or initiated updates. Table 2 gives the rule for each class of update. Despite the fact that the model showed that initiated and requested categories had different levels of attention, both were at a level (greater than 80%) that means that they should

always be presented. The default rules for presentation are therefore identical, and Table 2 combines these classes.

The generic rules are described in more detail below, and are followed by the refinements made to the presentation techniques that were applied to two highly interactive types of update that eye-tracking showed to have distinct patterns of user behaviour — auto-suggest lists and pop-up calendars. In these cases, the standard presentation rules are supplemented with extra interaction commands that help users interact with the information more effectively. Finally we describe how another phenomenon that became apparent through the eye-tracking study — visual bookmarking — was implemented. In each section, the output for the example content (see § 3) is given: these are the interactions experienced by participants in the evaluation (§ 6).

Update Class	**Cause**	
Effect	Requested or Initiated	Automatic
Insertion	1. Non-speech notification 2. Announce *"New content"* 3. Move focus to new content 4. Speak first chunk of new content	Non-speech notification
Removal	1. Non-speech notification 2. Announce *"Content Removed"*	Non-speech notification
Replacement	1. Non-speech notification 2. Announce *"Content replaced"* 3. Move focus to new content 4. Speak first chunk of new content	Non-speech notification
Rearrangement	1. Non-speech notification 2. Announce *"Content rearranged"* 3. Move focus to first moved element 4. Speak first chunk of moved content	Non-speech notification

Table 2: Summary of generic update presentation by category. Different sounds were used for notifying users of automatic and manual or requested updates. Some particular types of initiated update were handled in a more tailored way.

4.1. Notification

Nearly all types of update are initially presented to the user via a short non-speech sound. This makes the user aware that something has changed, but it is anticipated that such a background sound is no more or less disruptive than noticing a visual change in peripheral vision. Two sounds were used: one for

automatic and one for requested or initiated updates. The only situations where these sounds were not used were for pop-up calendars and auto-suggest lists. In these particular update types, the user was found to interact much more with the content than with other types (i.e., the information in the update nearly always affected the user's next action); the presentation of these updates is described in sections 4.7 and 4.8. It should be noted that the special user interfaces designed for these updates are not exceptions to the general rules, but are refinements, arrived at through further qualitative analysis of the data, that have been applied to common design patterns. These give interesting examples of the type of information that this approach yielded, and how further studies may enable refinement of audio user interfaces to other design patterns. Users had the ability to disable (and re-enable) notifications for updates.

4.2. Automatic Updates

While the sighted user studies showed that these were generally ignored by users (less than 25% of automatic updates were fixated), there are strong theoretical reasons for making the user aware that the update has occurred. Firstly, nearly a quarter were attended, so sighted users clearly thought that some were worth at least a glance. Second, navigating around a changing page is likely to be disorienting, particularly so if the page is not known to have changed. Finally, if screen reader users are to have an experience that is equal to that of sighted users, the same information should be available to them; this includes knowledge about page changes. For these reasons, the notification by non-speech sound was considered sufficient. Should the user wish to assess the value of the update, a command is provided to listen to the content without moving the focus.

The ticker was presented with the standard notification, regardless of the user's focus. If, however, the focus was on the ticker, the content was spoken on each update. If the user did not want to be notified of these updates, he or she could set the browser to ignore it.

4.3. Removals

Updates where content was removed from the page are announced with the non-speech sound, and the phrase 'content removed' is spoken. A command was available to allow users to hear the removed content.

4.4. Insertions

For user-requested or user-initiated insertions, the focus moves to the new content and the announcement *'new content'* is made. If the new content does not immediately follow the focus in the page map, however, this announcement is modified to *'moving to new content'*. Following the announcement, the first chunk of the new content is spoken.

The example of this type of content used in the evaluation was the expansion button that revealed instructions for the form (see § 3.4). When the user pressed enter while focused on the control, the instructions were revealed. The user

heard the beep, then *'new content'*, followed by the first chunk of information (*'Use this form... '*).

4.5. Replacements

For user-requested or user-initiated replacements, the situation is very similar to insertions. The announcements are modified slightly, however, to *'content replaced'*, if necessary followed by *'moving to new content'*. Thus, when selecting the special offers tab (§ 3.2), the user heard *'Content replaced. New Content. The Queens Hotel'*. Focus was left on the title, allowing the user to navigate on through the new content or to return to the control (see § 4.9. Similarly, when viewing the slides, pressing enter on the 'Next' control gave the output: *'Content replaced. Moving to new content. Blenheim Lodge'*. Again, the focus moved to the new content.

4.6. Rearrangements

Rearrangements are treated in the same way as replacements (it could be argued that they are actually just a special case of replacement): the announcement *'content rearranged'* is made, and focus moves to the first element of the rearranged content. When rearranging the results table by activating one of the table header controls (§ 3.8), the user therefore hears: *'Page rearranged'* followed by the name of the first hotel in the table, where the focus remains.

4.7. Auto-suggest Lists

These are a special category of user-initiated update, where a list of suggestions appears when the user types in an input box. The suggestions depend on the application but may give a list of common queries for a search box, or a list of matches where input is constrained (e.g., entering a destination). As updates are initiated by the user with a key press, our model shows that these updates are nearly always attended to by sighted users (95%). In addition to this basic fact, the sighted user studies also gave insight into the detail of how these updates were viewed (Brown et al., 2009). The results showed that suggestions further down the list were less likely to be viewed, with less than 50% of participants viewing any suggestions after the third in all instances encountered.

This deeper understanding of how these particular updates were used enabled them to be presented in a more carefully tailored manner. Auto-suggest list updates were identified by analysing the content of the update in those cases where the update has been initiated by the user typing in a text input box. If the content is found to be a table or a list, it is assumed that this is a list of suggestions. In this case, the first three suggestions are spoken automatically. If the user continues to type, or presses the escape button, this speech stops (new suggestions will be spoken if they appear). If the user wishes to select a suggestion, the enter button can be pressed before the next suggestion is spoken. If he or she wishes to browse the suggestions list, this can be done using the keys for next (or previous) sentence or word.

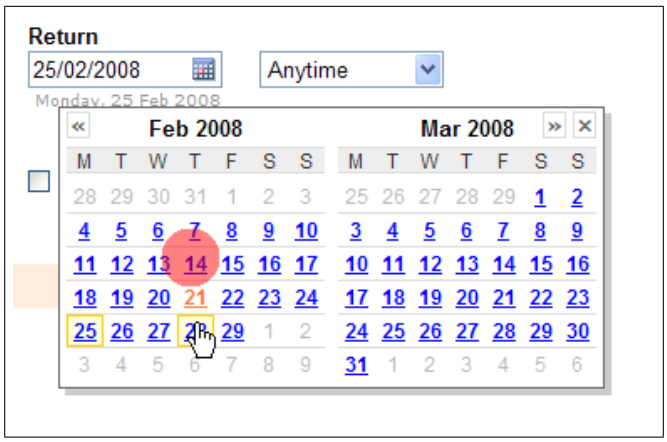


Figure 10: A participant selecting a return date from a calendar. The participant locates the departure date, and then moves down one row to select the date a week later.

4.8. Pop-up Calendars

Another category of user-initiated update is the pop-up calendar. Again, it was found in sighted user studies that these were nearly always fixated: of the 26 participants for whom pop-up calendars appeared, 14 requested it by clicking a button, all of these viewed it, and 12 initiated it by moving focus to the input field; all but 3 of these used the calendar, with 2 not fixating once. In addition, the eye-tracking data gave extra insight into *how* these updates were used, and hence why they were useful. In the study participants were asked to select a date of departure (for a holiday booking), then select a return date one week later. The eye-tracking revealed that presenting the calendar as a table (one week per row) made jumping a week at a time easy — users identified the date of departure, then looked down a column to find the date one week later (see Figure 10).

The non-visual implementation for pop-up calendars (Brown et al., 2010) used some simple heuristics to detect the update type, then employed the grid layout of the navigation keys to recreate some of the benefits of the table layout typical in these updates. An update is assumed to be a pop-up calendar if the update appears when entering a text field that has some characteristic of a date. Characteristics include containing text formatted as a date, or having a label, name, or id containing 'date'. Presentation involves a change of mode, and starts with the announcement 'Date Entry' followed by the date in the field (if no date is present, today's date is spoken). There is also a command to allow this mode to be entered manually. The user may then change the date a day, week, or month at a time, either forwards or backwards; this is done using the number pad keys, with the top row moving months, middle row moving weeks, and bottom moving days. The current day, date, or month is spoken using the

middle column, while the left and right columns move backwards and forwards in time respectively. The 'enter' button puts the selected date into the input field and the 'escape' button exits the calendar mode without modifying the input field.

4.9. Bookmarking

Another observation made during the sighted user studies was the way in which these users returned attention to controls. This was particularly the case where updates occurred through widgets such as a carousel. In this case it appeared that participants would glance at the new content without moving the mouse from the control; if the content was not of sufficient interest, the user could activate the control again with little further effort. In effect, the mouse pointer acts as a kind of bookmark, allowing easy return to the control.

In this implementation, the assumption (based on the user studies) that users will want to attend updates that they have requested means that the focus automatically moves away from the control when the update appears. This, coupled with the user's lack of peripheral awareness, can mean that the user loses awareness of where they are in the structure of the document (although the controls and widget content are typically neighbouring, this is not necessarily the case, e.g., when the controls are below the content). To compensate, to a limited extent, for this, our implementation automatically bookmarks those controls activated by the user. Thus, they can browse widgets such as carousels in a similar way to sighted users: press the 'next' button; assess the content; press the button, etc.

4.10. Reviewing

The system also gave users access to a list of updates, which gave them information including the type of update (removal, insertion, etc.) as well as the content.

5. Implementation

In order to test these rules, they were implemented in a self-voicing extension to the Firefox Web browser. This extension was based on the Fire Vox extension that enables exploration of Web pages using keyboard input and audio output. The evaluation is described in section 6; this section describes the implementation, explaining the detection and processing of updates, so that they could be presented according to the rules in sections 4.1 to 4.10.

The process can be split broadly into three:

1. Detection of page changes, then clustering them into meaningful updates.
2. Classification, according to the attributes of the update and the user's activity.
3. Presentation, applying the rules appropriate for the class.

Before describing these parts of the process in more detail, it is necessary to introduce the user interface. In particular we describe the methods for navigating around a page.

5.1. Navigation

The user interface for the experimental prototype was changed from that used in the standard Fire Vox, on the basis that this was not particularly intuitive, and might take more learning time than would be appropriate in a short (less than two hours) evaluation. The replacement interface was loosely based on that used in the Orca screen reader for the Gnome desktop on Linux. Navigating around a page can be done in two ways: sequential navigation, or element-based navigation.

Sequential navigation allows users to move from element to element as they appear in the Document Object Model (DOM) (a depth-first traversal of the tree). This can be done at three levels: element, sentence or word. Moving by element will read out the content of each HTML element from its start to either its end or a child element. For example, a simple paragraph element will be read in one go, but a paragraph containing a link will be read in three parts: the text up to the link, the text of the link element, then the text following the link. When moving a sentence at a time, sentences are determined from an element, so a sentence containing a link will also be split. Movement by word is, in principle, the same, although words rarely span element boundaries. Moving at these three levels of granularity is achieved using the number pad keys found on a typical desktop computer keyboard. Each row relates to a different level of granularity: the top row moves by element, the middle by sentence, and the bottom by word. The columns enable the user to move backwards (left column) or forwards (right column), or to query their current location (middle column).

Element-based (or structural) navigation is a method for jumping between elements of a certain type. For example, jumping between, and reading, the headings on a page can be used to give an overview of its content. This type of navigation is achieved by using character keys (e.g., 'H' moves between heading elements) when in structural navigation mode. This mode may be toggled between on, off, and automatic. In automatic mode, structural navigation is enabled unless the focus is in a form element requiring typing, such as an input box.

Links are announced as either 'external link', 'internal link', or 'control', the last of which describes links that activate a JavaScript function. Pressing enter on a link selects it, and is accompanied by a non-speech 'click' sound.

5.2. Detection

The aim of the first stage is to identify any changes to the document (Web page), and to group those which are related. Technically, the events that comprise an update, can be considered from three different viewpoints, those of the developer, the user, and the DOM. Consider the replacement of one paragraph with two new ones as an example. From the user's point of view, this is a single event: a straight replacement of the old content with the new (no intermediate stages are perceived). From the developer's point of view this might be coded as a two-stage process — remove the old paragraph, insert the new. From the point of view of the DOM, however, there are several events, one for the removal or addition of each element in the model. The aim of the detection and

clustering process is to identify model changes, and group them into units that would be perceived by a user as single events.

The basic method for detecting changes is to poll for changes. This is done every 1.5 seconds (a number achieved by trial and error). Firefox generates DOMMutationEvents when the DOM changes, which are listened for and noted. If any such events have occurred since the most recent poll, the update detection system generates a map of the model and adds this to a list of maps for the page. Comparing the new map with the last in the list identifies two sets of nodes: those that have been removed from the model and those that have been inserted into it. These two sets are passed to the clustering system, which groups nodes that are neighbouring in the model tree.

The sets of clustered nodes are then parsed to detect rearrangements and replacements. If there are neither insertions or removals, then the DOMMutationEvent was caused by nodes moving. In this case, the maps are analysed to determine the extent of this rearrangement — the node furthest down the model tree that contains all the moved nodes. In the other situation, where there are both insertions and removals, the map is analysed to determine whether these come from the same area. This is considered to be the case if the unchanged nodes before and after the removed chunk and inserted chunk are the same.

The end result of the detection, clustering, and grouping processes is a list of updates, which have been categorised as being one of: insertion, removal, replacement or rearrangement.

5.3. Classification

The aim of the second stage of the process is to classify updates so that the appropriate presentation rules may be applied. This is done by monitoring the user's activity, his or her location within the document, and the content of the update.

The model developed from the sighted user studies is shown in Figure 9. This shows that sighted users attended to updates with significantly different rates according to whether the update was initiated, requested, or automatic. The classification system in the implementation therefore uses the same categories. It also attempts to distinguish between keypress and click/hover initiated updates, although this is not simple, since users with visually impairments tend not to use a mouse. In addition, the classification considers the effect of the update: remove, insert, replace, or rearrange.

The first stage in the classification process, that of determining the effect of the update on the page, was performed in the detection and clustering process, and is described in section 5.2. The remaining task, therefore, is to determine how the update was initiated. This is done using the last action of the user, as shown in Table 3. Since this classification mechanism is based on heuristics, it uses the precautionary principle: if there is doubt, updates will be assigned to a more 'important' class (i.e., one which the model shows the user is more likely to attend).

To supplement the rules in Table 3, a list of 'live' regions is also stored. This records all regions of a page that have updated, and how that update

User's last action	**Inferred** update class (*cause* axis)
Activated a link or button whose target is a JavaScript function	Requested
Followed an internal link Entered a form field Exited a form field Submitted a form Typed in a text input field Modified an input field (e.g., radio button)	Initiated
Navigated around the page (other than those special cases above) Issued a browser command Nothing in the last X seconds	Automatic
Reloaded the page Followed an external link	New Page

Table 3: Update classification based on user activity.

was thought to have been initiated. This is used to manage potential conflicts on pages with several regions. On these pages it is possible to get automatic updates and requested or initiated updates coinciding, in which case using the above rules alone would lead to the automatic update being incorrectly classified. Monitoring live regions allows these conflicts to be resolved, by assuming that areas which have updated automatically always update automatically.

5.4. Future Work

There are many aspects of this implementation that make it sub-optimal for general use. The system described, however, was designed for proof-of-concept — primarily to test how effective the presentation rules derived from sighted user studies were for users with visually impairments. The following deficiencies are noted:

- Some of the messages are verbose, and would be better presented through non-speech sounds. The more verbose design was used as it requires less learning and, as such, is more suited to a 90 minute evaluation.
- The detection heuristics are relatively basic, particularly those distinguishing auto-suggest lists and pop-up calendars from other user-initiated updates. When used with known sites, however, they were sufficient to evaluate the rules.
- The use of the number pad keys is not an ideal solution (these keys are not always present, e.g., on laptops), but the interface offered appeared

appealingly simple and quick to learn, and under the conditions of the evaluations it was possible to ensure that this did not cause any problems.

6. Evaluation

The implementation described above — the 'SASWAT (Structured Accessibility Stream for Web 2.0 Access Technologies) browser' — tailors the presentation of dynamic content according to the user's activity, the focus of the user, and the nature of the content itself. In order to test whether this is a helpful approach, and to test the presentation rules applied to the different classes of update, an evaluation was performed. The core aim of the evaluation was to determine whether the interaction metaphors developed in the SASWAT project provided better access to dynamic micro-content than screen readers do at present.

6.1. Methodology

The study used a within-subjects design, to compare the presentation rules described above (§ 4) with a more homogeneous and less intrusive presentation style, similar to the current behaviour of many popular screen readers, including JAWS. This was done using two versions of an audio Web browser, named the 'SASWAT' browser and the 'base case' browser. These were identical apart from the manner in which they handled and presented updates. In the base case, the default method of handling updates was neither to announce the change nor to change the user's focus. The only occasions in which the focus was moved were when the focus node had been moved or removed from the document; in these cases, the focus moved to the nearest equivalent node. Thus, for the ticker, focus remained on the sentence, the latest content of which could be determined using the command for current chunk. For the table, focus moved to the top left cell (the user was not notified). Auto-suggest lists and pop-up calendars were both inaccessible to base-case users, who input their information without the additional content. The behaviour of the base case browser with respect to updates is similar to many popular screen readers.

The goal of the evaluation was to give participants an opportunity to compare the behaviour of both browsers whilst interacting with a variety of dynamic micro-content, so they could later specify which they preferred in a structured interview. As such, they completed the same tasks, on the same website, twice. To control for practice effects, half the participants completed the holiday booking tasks using the base case browser first, the other half using the SASWAT browser first. In addition to observing their behaviour and recording comments, participants were asked to rate the ease of use for each browser using a scale of 1 – 5, where 1 = very hard, and 5 = very easy.

6.1.1. Participants

12 participants were recruited for the experiment. P1m (participant codes reflect the order in which participants completed the evaluation and their gender) was a member of staff and p2m and p3m were students at the University

Participant	Frequency of Web browsing	Nature of Web browsing
P1m	daily	work
P2m	daily	study, email, facebook
P3f	daily	shopping, study, email
P4f	weekly	finding things out
P5f	daily	study, shopping, 'everything really'
P6m	daily	looking up historical information
P7m	only twice in college	N/A
P8f	daily	family tree, shopping (with help)
P9m	daily	work, study, email, occasional shopping
P10f	monthly	family tree (with help)
P11m	daily	shopping, price comparison, looking things up
P12f	daily	work, email, looking things up

Table 4: The frequency and nature of Web browsing of the evaluation participants.

of Manchester who were contacted via word of mouth (they had taken part in previous studies and expressed an interest in participating in future ones). These participants had provided feedback in informal iterative evaluation sessions during development so had had experience of using an earlier version of the SASWAT Web browser. The remaining participants were recruited through advertisements placed in Macclesfield and Stockport talking newspapers, and circulated by staff at Macclesfield Eye Society and Walthew House Deaf Blind Services in Stockport. P4f, p5f, p6m, p7m and p11m were service users and p8f, p9m and p10f were volunteers/staff at either Macclesfield Eye Society or Walthew House. P12f was an IT teacher who worked with blind students. Participants received an honorarium of £20 for taking part.

Two of the participants were partially sighted (p3f and p11m), five were registered blind with some residual vision (p4f, p6m, p8f, p10f and p12f) and five were profoundly blind. P1m, p2m, p3f, p6m and p12f were in the 20-45 age group; the remaining participants were over 45. All participants normally used Windows, and the majority browsed the Web using the screen reader JAWS (v. 10 or below). One participant used the screen magnifier ZoomText with audio (p3f). Two used ZoomText without audio, one of whom (p8f) had experience with audio, the other (p10f) didn't. P11m used a standard desktop computer set-up with a large font size, but had had experience of using JAWS at college. Thus, participants had a range of experience with screen readers, with slight bias towards the more experienced. Table 4 gives the experience of the participants.

6.1.2. Procedure

The investigator read out the Information Sheet and Consent Form, and when participants had indicated that they were happy with the information they had received, they signed the Consent Form. All participants signed the

form themselves, with the exception of p2m, for whom a second investigator signed on his behalf.

Participants were shown how to use the browser and given the chance to practice using the navigation commands on a shortened version of the Fire Vox User Manual home page. They then completed the tasks with each browser. Due to the step-wise nature of the interaction, the tasks were always completed in the same order. The participants were given their instructions verbally for each task as it arose. These told participants both what was required of them, and what sort of content they were about to encounter. The tasks were:

Ticker Move the focus to the ticker and listen to the latest news from the HCW Travel Company.

Tabs Select the 'Special Offers' tab. What is the name of the hotel featured in the first special offer? A second task requiring interaction with tabs occurred after the user had completed the slideshow task: Select the 'Book Now' tab, and within that select the 'Book Now' external link to go to the booking page.

Slideshow Find the locations featured in the next two special offers.

Expansion button Select the 'Instructions' control and read the instructions for using the form.

Auto-suggest list Enter 'Spain: Andalucia' in the 'Destination' text field.

Error messages Listen to the error message to identify the part of the form that contains an error. Note that this only applied to participants whose actions led to an error.

Table re-ordering What it is the name of the cheapest hotel? How many hotels are there with three stars?

In addition to voicing their thoughts, participants were able to ask questions and were given assistance whenever requested. Once the tasks were completed under both conditions participants took part in a structured verbal interview in which they gave the tasks difficulty ratings, and answered questions about their experience. This interview consisted of some introductory questions, then questions about the participant's experience with each type of dynamic content, finishing with a few more general questions. The introductory questions were:

1. Have you had your visual disability since birth?
2. Which assistive technology/screen reader do you usually use? Do you use Windows and Internet Explorer?
3. How often do you browse the Web?
4. What sort of things do you use the Web for?
5. How often do you use travel websites?

The following questions were asked for each type of content in the order in which they were encountered (ticker, tabs, slideshow, expansion button, auto-suggest list, calendars and additional content):

1. Have you come across *[type of dynamic content]* before on the Web?
2. On a scale of 1 to 5 (1 being very easy, 5 being very difficult), how easy was it to access the information provided by the *[type of dynamic content]* using the first browser?
3. And the second browser?
4. Would you like to be able to access information provided by *[type of dynamic content]* when you are browsing the web?
5. Can you think of a better way of presenting the information provided by the *[type of dynamic content]* in audio than the methods used here?

There was some variation according to the content: for tickers there was an additional question ('Did you find the non-verbal cue indicating that information was updating automatically useful?'); for auto-suggest lists, question 2 was only appropriate for the 'SASWAT' browser; question 2 was not appropriate for calendars. Finally, the following general questions were asked:

1. Overall, which assistive technology made it easiest to complete the booking task? Can you give a reason for this?
2. When the content on the page changes, is it useful to be told about this?
3. Do you think that if information on the web is available to someone sighted, it should also be available to someone who is visually impaired?
4. Do you have any other comments?

Each session was audio recorded and lasted between 50 and 105 minutes, with the majority being 60 to 70 minutes.

6.2. Results

In this section we present the quantitative results of the evaluation. The comments made by participants during the evaluation and subsequent interview demonstrate the reasons for their preferences and ratings, and are thus discussed in section 7. The quantitative data were obtained from the difficulty ratings given by participants, which were recorded for all types of dynamic content except for ASLs and Calendars. In these cases, the base case scenarios required users to type their information into the input box without any interaction with dynamic content, so scores were not recorded. The median ratings for each type of content under each condition are given in Table 5. Some participants gave intermediate scores (e.g., 3.5); in these cases the score was allocated evenly between the two values.

Of the 63 pairs of ratings given (one user rating the same content under each condition), in only 1 case was the base case presentation rated easier than the SASWAT (p5f rated the ease of using tabs as 2.5 for SASWAT and 3 for the base case). In 4 cases equal ratings were given (all for the expansion button),

while for the remaining 58 (93%) the presentation of the SASWAT browser was preferred.

Analysis of the ticker, tabs, slideshow, expansion button and table scores using the Wilcoxon Matched-Pairs Signed-Ranks Test shows that ratings are significantly higher for the SASWAT browser ($p < 0.01$). No differences were found between the results of those participants who were experienced with screen readers and those that were less experienced (p3f. p8f, p10f, p11m).

7. Discussion

The quantitative data showed that for tickers, tabs, slide-shows, expansion buttons, error messages and rearranging tables, the majority of participants preferred the access provided by the SASWAT browser. The behaviour of the users, and the comments and feedback provided by them support this conclusion, and extend it to the ASLs and calendars. They also provide insight into why users liked the SASWAT form of presentation, and how effective the technique of using eye-tracking to guide development is.

A key reason for their preference for the tailored presentation was the immediate feedback the browser provided when content had updated. All participants expressed a desire to be told when content on the page had changed, and all thought that a facility like the beep alerting the user to the updating ticker would be useful. Whilst being notified of automatically updating content was desirable, receiving feedback about a requested update was vital. When using the base case browser, which did not provide verbal feedback, participants assumed that the control had not worked.

It was not only the fact that feedback had occurred, but the fact that it was informative that appealed: p11m, for example, appreciated the reassurance provided by the SASWAT browser: '[the SASWAT browser] explained what was happening... It's alright doing it, and you've got to rely on the fact that it's done it. But on [the SASWAT browser] it told you that it had done it, so I was quite confident that we were where we should be.' The fact that the SASWAT browser

Dynamic content	**SASWAT**	**Base case**	**Wilcoxon test**
Ticker	4	3	$N = 12, W = 78, p < 0.001$
Tabs	5	3	$N = 12, W = 76, p < 0.001$
Slideshow	5	3	$N = 11, W = 66, p < 0.001$
Expansion button	4	4	$N = 8, W = 36, p < 0.01$
Error Messages	5	1	N/A ($N = 4$)
Table	5	3	$N = 11, W = 66, p < 0.001$

Table 5: Median scores for ease of access to each type of dynamic micro-content with the base case and SASWAT browsers (1 = very difficult; 5 = very easy). The Wilcoxon Matched-Pairs Signed-Ranks Test was used to test for significance, except for Error Messages.

moved straight to and read the new content was also viewed very positively by the participants – indeed, it was what they expected to happen. When p4f failed to receive feedback after updating the slideshow with the base case browser she said, 'It's just, I thought it was going to tell me... I was just wondering what the next offer was.'

Participants liked the fact that they got to the information they wanted quickly. P9m preferred the SASWAT browser because 'it gave you information more immediately, and... yeah, that's why really... It's about quick accessibility for me really – you do the job as quickly as you can.' P12f also felt the SASWAT browser provided a quick, intuitive response to a change in dynamic micro-content: 'You've got a lot more control with something like this than what you would have normally, just with JAWS. It's good... it's a lot easier to follow. It's a lot easier to use. It doesn't stop talking to you. It's not inconsistent – it's consistent with its information... Once I'd had time to play I feel I could get quite competent with that.'

When using the SASWAT browser, participants felt confident they knew where they were on the page. In the words of p8f: 'It's just really, each time you go on to each page it actually speaks to you. It tells you where you are and what part of that page you're up to. Because you're having to picture it in your mind. Where it is. And it's very important that you know where you are on each sheet or wherever.' Most of the participants experienced difficulties with orientation and navigation when using the base case browser, and these were particularly evident in the slideshow task. The majority of participants (9 of the 12) navigated the wrong way (forwards, rather than backwards) when attempting to reach the new content, even though they were aware that the focus was still on the 'Next' control, and had previously been informed that this was at the bottom of the slide. When using the SASWAT browser, participants intuitively navigated forwards and quickly reached the content they were looking for. The automatic bookmarking feature was used at some point by all participants to return to the controls once the content had been reviewed. We believe that this functionality is a crucial part of the system if the user's focus is to be moved automatically, as it means that they do not need to manually navigate back to the controls once they have finished reading the new content. The effective use of automatic focus changes is contrary to the expectations of Borodin et al., and perhaps would be counter-intuitive if it were not for the eye-tracking data. Users found it more disorienting to remain where they were than to have their focus move. This was despite the fact that they had been informed about the type of content and were aware of the effect their actions would have.

It is interesting to note that the one content type where there was no significant improvement in the tailored case is expansion buttons — in this case it can be seen that shows that this was not due to poor performance of the SASWAT browser (5 people rated it with a score of 5, and a further 5 with a score of 4), but primarily because participants found these updates relatively easy to handle in the base case (the median score was 4). This is likely to be due to the fact that the update is inserted into the content in a linear fashion, i.e.,

immediately after the control. For most of the other types of dynamic content, the interaction is more complex, and often less 'linear', and in these cases, the users found the SASWAT browser more helpful.

This type of behaviour provides further evidence for the efficacy of this approach. The tailoring, which actually leads to a wider range of behaviours by the browser — feedback and focus move differently for different types of update — was perceived as being more consistent by the users. A likely explanation for this is that presenting updates according to class gives behaviour that is more consistent with the user's mental model of his or her interaction with the page: the potentially confusing behaviour of moving the focus actually keeps a better match between where focus really lies on the page and where the user thinks the focus is in his or her mental model of the page.

The automatic updates provide an illuminating contrast. The ticker (which spoke the new content when it was the element of focus, and gave notifications otherwise) was found annoying by most participants. All bar one chose to turn off notification for this item, and three participants commented on the notification being annoying. While we believe users should be informed of these types of update, this needs to be done as unobtrusively as possible, and have the facility for being turned off. In Hearsay Dynamo , automatic updates were essentially presented in the same way as in the SASWAT browser. The two systems differ, however, in the way they handle other updates. While Hearsay Dynamo does not differentiate between update types, the SASWAT browser does, and the conclusions of this evaluation are that this differentiation aligns better with user's expectations and mental models.

Another measure of the effectiveness of the update presentation in the SASWAT browser is the number of errors made. In the base case, all participants except p12f made an error in completing the form, and were presented with an error message when the form was submitted; no such errors were made by users of the SASWAT browser. The errors made were a combination of incorrect date entry and invalid destinations (e.g., the destination did not match one in the database), due to typing errors and the inability to browse the auto-suggest list. Errors were also more common for users of the base case when interacting with other types of dynamic content (e.g., the slide-show navigation discussed above), although these generally resulted in disorientation, increasing the time taken to complete the tasks (time to task completion was not measured, as participants were free to make comments and ask questions during the evaluation, meaning that overall time would not necessarily reflect the difficulties encountered).

8. Conclusions

The eye-tracking studies, iterative implementation, and final evaluation of this tool all confirm that informing users about updates is important, but suggest they do not consider them to all have equal importance. We recommend presenting automatic updates as unobtrusively as possible, but that user-initiated updates should be presented more directly. Concerns that automatically moving the user's focus to present new information would cause

disorientation do not appear to have been borne out. Indeed, moving focus to new content seemed to result in a better match with the user's mental model. These findings could be applied by screen readers as a default way of handling updates that do not have WAI-ARIA markup and to inform developers implementing these tags. The more detailed investigation of auto-suggest lists and pop-up calendars demonstrate that this approach can yield information that helps build effective audio interfaces.

This research has extended and, to a certain extent, confirmed the work Borodin et al. (2008) did with their HearSay Dynamo browser. While they found that users benefited from update notification, they examined a limited range of updates, and used the same form of notification for each. The results presented here broadly confirm that users find notification helpful, and extend this conclusion to a wide range of update types, including pop-up calendars, auto-suggest lists and slide-shows; in each case users found notification helpful. This work further extends HearSay Dynamo, however, by tailoring the notification according to the update type. Thus automatic updates are presented very tersely, while following user-requested and initiated updates, the user's focus is moved to the new content and the system starts to read it. The most substantial departure from the HearSay Dynamo work, however, is that we have shown that moving the focus of the user, for certain types of update, does not lead to the disorientation that Borodin *et al* predicted. Implementation of such a feature does, we believe, necessitate the kind of automatic bookmarking implemented in this system, so that the user is able to review new content then simply return to their original location; without this we would expect orientation problems.

Because potential user fatigue meant it was inappropriate to test both the use of notification over a wider range of updates, and to compare different methods of presentation, evidence for the benefits of the presentation techniques proposed here are qualitative. The fact that users were aware that changes were going to occur, however, means that it is possible to draw some valid conclusions, primarily that moving the focus and speaking the new content for user-initiated and user-requested updates reduces disorientation. In fact, we suspect that the benefits of HearSay Dynamo may lie more in the way it allows users to jump to new content than in the fact that it notifies them of a change. Our study suggests that it is not sufficient for users to know that an action will cause a change, or even that a change has occurred, but that the change needs to be presented in a way that prevents disorientation. Further studies, comparing different methods of notification and, crucially, comparing methods of access to the new content, are necessary to confirm this, and to test further refinements to update notification.

A second limitation of this study is that it does not directly compare tailored presentation against uniform presentation of updates. Nevertheless, we believe that the study supports the use of tailored presentation. The use of simple, unobtrusive, notifications for automatic updates is supported by the fact that all but one of the participants chose to turn these notifications off — demonstrating that they were of minimal interest. It is also clear that moving the focus to the updated content, which was found to be effective for user-initiated updates,

would prove very disruptive if applied to automatic updates — if this technique is to be used, tailoring is essential. Finally, it should be noted that the dynamic interaction required by updates such as auto-suggest lists would be very difficult to achieve using a system that notifies with the option to then read the new content.

The work of Hailpern et al. (2009) with their DTorial system, a help system for visually impaired users of dynamic Web applications, shows that the technique applied to user-initiated updates here (moving the focus and speaking) has been applied to dynamic content in another context, and liked by users. Their system dynamically injected tutorials into appropriate parts of a complex Web application (GMail), and automatically moved the user to the new content, and was generally found to be effective by participants in their evaluation. This technique fits with our model of how sighted users might interact with such information, and with how our participants successfully coped with similar focus changes. It also suggests interesting future work, as the experiences of participants in our evaluation suggested that knowledge of the type of content they were about to interact with was helpful; implementing a tutorial system such as DTorial that could be injected into any web page to provide help for more general types of interaction (such as tabs or slide shows) could be useful for novice Web users who are using screen readers.

Overall, the SASWAT browser received a very positive reception from the evaluation participants. Qualitative analysis of the results shows that this may because the SASWAT browser replicates for visually impaired users some of the functionality that dynamic-micro content affords sighted users. Whilst this study cannot prove that the approach – tailoring presentation of updates using audio mappings based on observations of sighted users – provides screen reader users with optimal access to dynamic micro-content, it certainly indicates it can offer a significant improvement on the access available through current assistive technology.

In summary, our experiments and analysis lead us to the make following recommendations:

- Browsers should support both ARIA and DOMMutationEvents.
- Developers should implement and test ARIA markup for dynamic content. Approaching the design in the same way as we did for auto-suggest lists and calendars — providing the benefits of the visual implementation, rather than a direct translation of it — could prove beneficial.
- Developers should consider the structure of their page so that new content appears in a location that will appear logical to screen reader users as well as those viewing a page rendered with the standard style (i.e., when the DOM is traversed depth-first). For example, it will reduce the possibility of disorientation if, when a control is selected that causes new content to appear, new content is inserted into the DOM tree as soon as possible after the control.

- Screen readers should notify users whenever a page changes, unless the user has turned off notification.
- Where ARIA markup has not been used, screen readers should use simple heuristics to determine the type of update and present it accordingly:
- – Automatic updates should be presented in an unobtrusive manner.
  – For user-initiated and requested updates, the focus should move to the new content (with appropriate warning).
  – Users should always be able easily to return to their location at the time of the update.

We believe that improving access for screen reader users to dynamic content is necessary but possible. If the recommendations above are followed, we see no reason why these users can't use dynamic content in a way that enhances their use of the Web rather than acting as a barrier to it.

9. Acknowledgements

This work is part of the Single Structured Accessibility Stream for Web 2.0 Access Technologies (SASWAT) project and is funded by the UK EPSRC (EP/E062954/1). As such the authors would like to thank them for their continued support.

References

Alliance for Technology Access, 2000. Computer and Web Resources for People with Disabilities: A Guide to Exploring Today's Assistive Technologies, 3rd Edition. Hunter House, iSBN: 978-089-79330-01.

Borodin, Y., Bigham, J. P., Raman, R., Ramakrishnan, I., 2008. What's new? — making web page updates accessible. In: Assets '08: Proceedings of the 10th international ACM SIGACCESS conference on Computers and accessibility. ACM, pp. 145–152.

Brown, A., Jay, C., 2008. A review of assistive technologies: Can users access dynamically updating information? Technical Report, University of Manchester, `http://hcw-eprints.cs.man.ac.uk/70/`.
URL `http://hcw-eprints.cs.man.ac.uk/70/`

Brown, A., Jay, C., Harper, S., 2009. Audio representation of auto suggest lists. In: W4A'09: Proceedings of the 2009 Cross-Disciplinary Conference on Web Accessibility (W4A). pp. 58–61.

Brown, A., Jay, C., Harper, S., 2010. Audio access to calendars. In: W4A'10: Proceedings of the 2010 Cross-Disciplinary Conference on Web Accessibility (W4A).

Burks, M. R., Lauke, P. H., Thatcher, J., Rutter, R., Waddell, C., 2006. Web Accessibility: Web Standards and Regulatory Compliance. Friends Of Ed.

Carmi, R., Itti, L., 2006. Visual causes versus correlates of attention selection in dynamic scenes. Vision Research 46, 4333–4345.

Chen, C., 2006. CLC-4-TTS and Fire Vox: Enabling the visually impaired to surf the internet. The University of Texas at Austin Undergraduate Research Journal 5, 32–42.

Gibson, B., 2007. Enabling an accessible web 2.0. In: W4A '07: Proceedings of the 2007 international cross-disciplinary conference on Web accessibility (W4A). ACM, New York, NY, USA, pp. 1–6.

Hailpern, J., Reid, L., Boardman, R., 2009. DTorial: An interactive tutorial framework for blind users in a web 2.0 world. In: Gross, T., Gulliksen, J., Kotzé, P., Oestreicher, L., Palanque, P., Prates, R., Winckler, M. (Eds.), Human-Computer Interaction INTERACT 2009. Vol. 5726 of Lecture Notes in Computer Science. Springer Berlin / Heidelberg, pp. 5–18.

Jay, C., Brown, A., 2008. User review document: Results of initial sighted and visually disabled user investigations. Technical Report, University of Manchester, `http://hcw-eprints.cs.man.ac.uk/49/`.
URL `http://hcw-eprints.cs.man.ac.uk/49/`

Keith, J., 2006. Hijax: Progressive enhancement with ajax. In: Proceedings of X Tech 2006, Building Web 2.0.

Mahemoff, M., 2006. Ajax Design Patterns. O'Reilly Media, Inc.

Oreilly, T., 2007. What is web 2.0: Design patterns and business models for the next generation of software. Communications and Strategies (65), 17–37.

Parkhursta, D., Law, K., Niebur, E., January 2002. Modeling the role of salience in the allocation of overt visual attention. Vision Research 42 (1), 107–123.

Raman, T. V., September 2008. Specialized browsers. In: Harper, S., Yesilada, Y. (Eds.), Web Accessibility: A Foundation for Research, 1st Edition. Human-computer Interaction Series. Springer-Verlag, Ch. 12, pp. 195–213, iSBN: 978-1-84800-049-0.

Scaife, M., Rogers, Y., 1996. External cognition: How do graphical representations work? International Journal of Human-Computer Studies 45 (2), 185–213.

Thiessen, P., Chen, C., 2007. Ajax live regions: chat as a case example. In: W4A '07: Proceedings of the 2007 international cross-disciplinary conference on Web accessibility (W4A). ACM, New York, NY, USA, pp. 7–14.

Thiessen, P., Chen, C., 2009. ARIA live regions: An introduction to channels. Journal of Access Services 6 (1), 215–230.

Thiessen, P., Russell, E., 2009. WAI-ARIA live regions and channels: ReefChat as a case example. Disability & Rehabilitation: Assistive Technology 4 (4), 276–287.

Zajicek, M., 2007. Web 2.0: hype or happiness? In: W4A '07: Proceedings of the 2007 international cross-disciplinary conference on Web accessibility (W4A). ACM, New York, NY, USA, pp. 35–39.