Achieving Accessible Rich Internet Applications

Linn Steen-Hansen[1]
Siri Fagernes[2]
[1]Agency for Public Management and eGovernment (Difi)
[2]Westerdals Oslo ACT

Abstract

This work presents guidelines on making accessible Rich Internet Applications (RIAs). The guidelines have been developed based on a thorough literature review, summarizing the current accessibility issues related to RIA. Existing solutions and recommendations have been processed and converted into a set of guidelines, aimed at those responsible for developing RIA solutions. Participants working with web development and accessibility evaluated the guidelines. The most important contribution from this project to the field of web accessibility research is strong indications of a need for process oriented accessibility guidelines.

1 Introduction

A Rich Internet Application (RIA) is a web application designed to resemble regular desktop applications. A RIA normally runs inside a browser and usually does not require any client-side software installation [1]. This approach allows the client system to handle local activities, calculations, reformatting and so forth, thereby reducing the overhead of client-server traffic. Most RIA development is based on standardized technologies like HTML, CSS and JavaScript [2].

RIAs increasingly rely on client-side code execution. New content can be obtained using JavaScript/AJAX, without refreshing or loading a new web page. User interaction can modify visible elements on a web page without requesting data from a server. Alternatively, an AJAX request to the server can fully modify the displayed content. In both cases, a new version of a page will be available without changing the URI [3].

JavaScript is a lightweight scripting language that has become increasingly popular. Currently, it is the most used front-end programming language on the Web, and 94 % of the top 10,000 websites visited use it [4]. Nevertheless, recent research indicates that less than half of modern web applications are accessible to people using *screen readers* [5], and that use of JavaScript and client-side code execution can decrease accessibility for people with disabilities in general [6], [7], [8], [9].

This project aims at making it easier to create accessible RIAs using existing standardized technologies. More specifically, we focus on the following research questions:

This paper was presented at the NIK-2015 conference; see http://www.nik.no/.

RQ 1: What specific problems exist with RIA accessibility?

RQ 2: What can be done to avoid RIA accessibility problems?

2 Web accessibility issues

The nature of the Web has changed dramatically over the years. From resembling physical documents conveying static information, the Web has become increasingly interactive and dynamic [6]. While the Web has become more technologically advanced, websites (and IT systems in general) now play a crucial role in our society and daily life [10]. The Web can simplify our recreational activities and aid us in our mandatory tasks associated with our duties as citizens. In addition, software programs have become indispensable in our professional as well as educational environments. Software programs, websites and applications targeted at a broad audience should be usable for *all* target users, which means people of all ages and backgrounds, as well as physical and cognitive abilities. Statistics from the World Health Organization (WHO) reveal that 12 % of the population in high-income countries has some kind of disability [11]. This is a user group of significant size, and not a homogenous one. It grows as the population grows older, and issues vary in severity. Further, people with disabilities is not just *one* particular segment. They exist in all segments and are just as diverse and different as able-bodied people.

The *World Wide Web Consortium* (W3C) defines Web accessibility as follows:

> Web accessibility means that people with disabilities can perceive, understand, navigate, and interact with the Web, and that they can contribute to the Web.

Accessible websites can also be a direct result of good design. Achieving good design of websites requires understanding some core matters; what you are designing for and the purpose of your site, your users' needs, and what visitors really wish to achieve using your site.

Accessibility can be seen as a *subset* of usability. Accessibility is a continuum and changes according to the technology we depend on. However, for users of assistive technology (AT) there are some core issues for each specific user group that does not change even if the technology does. As an example, blind users need to be able to access equivalent content to images, and people with limited physical mobility will benefit from a minimal amount of links to tab through [9].

3 Methodology

The goal of this project was to start the process of creating an understandable and easily applicable set of guidelines for developing accessible RIAs. This tool should be easy to grasp and to apply during a development process. It should have a clear focus on issues concerning dynamic and interactive applications. There is a particular focus on how to *bring accessibility into the development process*. A developer knowing little (or nothing) about accessibility should be able to quickly get an overview and understand what is needed to secure accessibility within this scope. The expected outcome of this project was an in-depth theoretical knowledge of how to create accessible RIAs using standardized technologies.

The methodology has been divided into five parts:

1. Surveying the existing research literature, studying RIA accessibility issues that have been discovered so far.
2. Surveying existing suggestions for solutions to current RIA accessibility issues.
3. Creation of guidelines aimed at smoothing the process of RIA accessibility.
4. Evaluation of guidelines to examine what works and what does not work.
5. Modifying the guidelines, based on the results from the evaluation.

Studying accessibility issues and solutions

A literature survey was conducted to discover problems with RIA accessibility. Based on the findings in the existing literature, a distinction was made between:

- *Social issues:* e.g. attitude towards accessibility among web developers, managers, customers and other stakeholders.
- *Tool issues:* problems with automatic testing tools and guidelines.
- *Issues directly linked to RIAs:* issues concerned with the dynamic and interactive nature of these applications and the problems this causes for users interacting with an interface.

The literature search was performed searching the following scientific databases: ACM Digital Library, IEEE Xplore, ScienceDirect, SciFinder and Web of Science. To also cover additional relevant material not accessible through these sources, the search was also performed in Google Scholar and Google. Search queries include:

- Rich internet applications AND accessibility
- Rich internet applications AND accessibility AND challenges
- Rich internet applications AND accessibility AND solutions
- HTML5 AND accessibility
- JavaScript AND accessibility

In addition, the names of renovned accessibility experts (and authors discovered in the search process) were used as search criteria.

Creating guidelines

The process of creating the guidelines began with sorting the material from the literature study and presenting it in a way that is helpful for development teams working with accessibility. This means answering the following questions:

1. *Which* requirements need to be fulfilled for a RIA to be (considered) accessible?
2. *How* do we assure that these requirements are fulfilled?

The literature survey pinpointed problems with updates not being detected, non-existing semantics, problems with keyboard navigation, pop-up windows and over-engineered interfaces as typical RIA issues. However, there was also broad agreement among scholars and the IT industry of that it should be possible to solve these problems. Some examples of solutions reappearing in the literature are using WAI-ARIA, using progressive enhancement and unobtrusive JavaScript, test extensively and think accessibility from the beginning of a project. These suggestions formed the basis for the guidelines.

There was a focus on making the guidelines as specific as possible. The material was first sorted into two main themes; *technology* and *process*. Further, the material was categorized based on *what* is recommended, *how* to execute the recommendations, *why* it is recommended and finally, *who* recommends it. The first two questions focus on specific advise and how to implement these suggestions and are placed in the guidelines. The second two questions are more concerned with the validity of the advise. It was considered to provide explanations of validity within the guidelines. However, to keep the guidelines as short and readable as possible, this was left out in the first version.

Evaluating and updating the guidelines

Due to time limitations, this project focused on evaluating the usability of the guidelines and not their reliability when it comes to ensuring accessibility. The participants were four developers with different levels of accessibility knowledge, and one accessibility expert. All participants were used to working with guidelines. The participants were sent the guidelines some weeks before the interviews. To get as much as possible out of the participants, the interviews was modified to simply allowing the participants to reflect on specific questions while reviewing the guidelines, before the interviews took place. It was expected that this would result in deeper reflections and expressed thoughts that could be discussed in more detail during the interviews. This strategy proved successful, as it resulted in natural conversations about the guidelines and what reflections each participant had made in the beginning of each interview before moving on to the interview guide. The discussions touched upon many of the planned questions, and in most cases the interview guide served as a way to summarize and wrap up the conversation and make sure everything was covered. Based on the interviews a list of suggestions for changes was prepared. What changes to make when creating a new version of the guidelines were chosen from this list, based on what were considered critical weaknesses and doable within the scope of this project.

4 The guidelines

In this section the guidelines are presented, along with some explanations and reflection justifying each guideline. The full version is available online at `http://accessibilityagent.no/guidelines`.

Have accessibility expertise on the team.

To use tools and guidelines reliably and effectively and successfully evaluate or develop accessible websites and applications, accessibility expertise may be required. This has to do with the complexity of accessibility issues [12], [13], [14], [15].

Include accessibility from the beginning of a project.

Testing and finding workarounds for accessibility can be more time-consuming than designing for accessibility from the beginning [14]. Having accessibility as an integral aspect of the design process and not an add-on or separate activity will improve the

quality of the end result [16]. Hence it is recommended to introduce accessibility from the beginning of a project and make it a priority throughout the development process.

Communicate accessibility within the team.

It is important that *everyone* on the development team understands that accessibility is interdisciplinary, *why* accessibility is important and *how* it affects different sides of web development. Exactly how accessibility is communicated may be different for each team and each project. An introduction course or a seminar could be useful. Seeing someone with a disability test the system under development might be an eye-opener regarding the user perspective. It could increase knowledge of how to enhance the experience of these users while at the same time highlight the human aspect of accessibility. To make sure decisions made on the basis of accessibility is maintained, it is useful to explain this to the programmers so it is not overlooked in the implementation.

Test for accessibility.

Testing is fundamental in any development process. It is good practice to test for accessibility on an equal basis as usability or user experience. It is essential to start testing early in the process [14]. Testing with *one* user early is better than testing 50 near the end [17]. It is recommended to continue verifying accessibility at key stages and testing throughout the process (W3C, 2011). The more often testing is done, the more certain developers can be of the quality of their product [9].

Use existing principles for good design.

Creating accessible interfaces really boils down to good design practice, well-formed code, solid information architecture, accessible form validation and error recovery. This will ground the designs in best practices and therefore bring extra benefits such as more future-proof, interoperable applications and websites [9].

Use WAI-ARIA mark-up.

WAI-ARIA is an accessibility-focused language that can bridge the semantic accessibility gap in projects [9]. Studies have shown that WAI- ARIA enhances accessibility [1]. WAI-ARIA supplies an application with the semantic metadata it needs to communicate well with AT, enabling user agents and browsers to understand the semantic mark-up [6], [18], [8].

WAI-ARIA also provides semantics that currently does not exist in other languages. One example is the properties connected to *live regions*, that ensure that updates are detected by AT. Notification based on a couple of simple rules can save considerable confusion because WAI-ARIA can alert the user if AJAX has updated any part of the page [14].

Follow the HTML5 standard.

Using HTML5 is recommended because its rich semantics creates a good structure. It also offers advanced input types and well-functioning integration with scripts [18]. Using HTML5 has some of the same benefits as adding WAI-ARIA [9]. Research show that websites made with HTML5 often are more accessible than websites developed based on earlier standards [19].

Combine WAI-ARIA and HTML5.

Many WAI-ARIA roles are similar to HTML5 elements. Using these in conjunction is a way of making applications backwards compatible with legacy versions of AT. WAI-ARIA support is a part of the HTML5 specification and should therefore validate without problems. Previous HTML versions does not validate with WAI-ARIA. There are also certain correspondences between the two specifications in the HTML5 spec and some elements have built-in WAI-ARIA roles [20].

Use Progressive enhancement.

Progressive enhancement is a widely used method for creating web applications that works on a variety of devices, user agents and connection speeds. It emphasizes accessibility, semantic HTML mark-up, external style sheets and scripting technologies, and uses web technologies in a layered fashion (Buckler, 2009). The content is marked up by well-structured HTML, correctly labelled forms, descriptive alternative text and so on. A separate CSS file is added for presentation. Finally, JavaScript is added to aspects of the site to enhance functionality to user agents that can handle it correctly.

Progressive enhancement benefits users of AT because the basic content will always be available and independent of the scripting to work. Also, it makes sure the code does not break user agents that cannot understand JavaScript. In addition it ensures keyboard accessibility as long as device-independent JavaScript methods are used. Progressive enhancement also ensures backward compatibility with older browsers and AT. This may solve some of the problems with use of old versions of browsers or equipment and future proofing of technology [9]. In addition it is useful when it comes to scalability, performance and maintenance [21].

Use unobtrusive JavaScript.

Unobtrusive JavaScript is particularly relevant for people whose browser lack JavaScript support, or if JavaScript fails. Using JavaScript unobtrusively means only using it when absolutely necessary and not making it a requirement for a functional application [22]. JavaScript is only added to enhance the user experience, make the application more responsive to user needs and provide quick access to information. It is removed from the HTML and tightens the separation between structure, presentation and behavior [21].

Unobtrusive JavaScript also means avoiding unnecessary movements on a web page, unintuitive widget functionality and unfamiliar controls. It aims for giving users a more seamless experience. Good technology should be invisible. Users will notice unwanted behaviors and especially if something does not work as expected [9].

Use device-independent methods.

When used correctly to add behavior to an object, *device-independent* JavaScript methods should ensure accessible interaction with any input device from eye-tracking and scanning software to keyboard, mouse and touch (WebAim 2015). Use of device-independent methods is a simple and powerful example of a more accessible way to add JavaScript to content because it allows for making some events keyboard accessible through JavaScript alone, not having to add WAI-ARIA elements [9].

Use accessible modal windows instead of pop-ups.

Modal windows are in many ways better and more usable than pop-up windows. They open inside the webpage instead of in a new window and saves the trouble of dealing with multiple windows. Modal windows do not hide behind the browser windows, which could easily make the users miss them entirely. They grab attention, darken the background and allows users to effectively focus on the tasks in the modal window. In addition they match the website, which makes them look connected to the site and seem less like pop-up ads. This seems safer and more secure. Creating a modal dialog that is fully accessible with WAI-ARIA requires some work, but is achievable (Zehe, 2015). Nevertheless, modal windows should only be used when the purpose is to completely take the user's focus and attention off the browser window to the modal window (UX Movement, 2011).

Use technology that facilitates accessibility.

Technology that facilitates accessibility should in theory make it easier to quickly ensure accessibility without applying time-consuming tools and guidelines, doing extensive accessibility testing, or even knowing much about accessibility. However, this technology is far from perfect at this point.

Toolkits and frameworks

Toolkits and frameworks enable faster and easier Web 2.0 application development. Some researchers believe the immediate challenge related to accessibility and RIA is building Web 2.0 development frameworks and technologies with a greater degree of WAI-ARIA support so that developers do not have to be experts in what constitutes accessibility. The efficiencies of accessible widget libraries and platforms cannot be underestimated, mainly because they often have already conquered issues between platforms and browsers. Although using these frameworks can be a great benefit, it is important to accurately wire and set the properties of accessible UI widgets [14], [1].

Web components

Web components facilitate accessibility because they can define widgets with a level of visual richness and interactivity not possible with CSS alone. It also offers an ease of composition and reuse not possible with script libraries today. They can be used to create custom elements and extend existing HTML elements with built-in behaviors. Web components are well supported in AT. Screen readers can access content in Shadow DOM without problems and the Shadow DOM is navigable by keyboard [20], (Sutton, 2014).

However, Web components will only be as accessible as they are designed to be. It is essential that developers establish a component's accessibility, and make changes if needed. Because anyone can make a Web component, it is likely that the quality will be low on many of them. WAI-ARIA information and tabindex should be added to custom elements. Given that the point of Web components is to add new types of objects, via style and scripts that are not yet directly supported by the language of the page, applying WAI-ARIA is very appropriate [20].

5 Results from the evaluation

This chapter presents the positive and negative feedback from the interviews and some of the suggestions for changes to the guidelines.

Positive feedback

The guidelines were viewed as useful, understandable, reliable and more manageable than existing guidelines. It was believed that they could contribute to build accessibility competence and ensure more accessible applications. Especially the process-oriented guidelines were very much appreciated and viewed as an excellent project methodology.

Suggestions for changes

Several suggestions for changes to the guidelines were made. Many were quite detailed and straight forward: providing an introduction to the guidelines, highlighting the exceptional qualities of WAI-ARIA and also to comment specific problems with WAI-ARIA. This article will in the following focus on the suggestions for changes that are most interesting to discuss and may reveal some underlying perceptions of accessibility.

Explain how guidelines benefit accessibility

It was suggested to have a section under each guideline, explaining how this guideline will contribute to accessibility. Tools should promote knowledge transfer and deepen the understanding of accessibility [14]. It was believed that explanations would increase the guidelines' usefulness and there is reason to believe that this will increase the level of learning and ability to build accessibility competence. This also makes it easier to explain to e. g. clients and management why one is doing something. It is also more difficult not to do something if its merits are known. These findings support previous studies of WCAG showing that when developers do not understand why they should use a guideline, the motivation to apply it decreases [23].

Prioritize and time estimate the guidelines

It was chosen to not indicate any priority ranking of each guideline, as the guidelines have not yet been tested for reliability. This means it cannot be established if some are more important that others, and if so, which ones. Based on the feedback from the participants in this study, priority information should be included. The potential benefit of giving priority information, is that it stops developers from just doing the easy parts and quick fixes and ignore the more complex issues.

On the other hand, giving a prioritized list could also be interpreted as some things do not really need to be done. In addition, prioritizing these guidelines may prove difficult because of the comprehensiveness of many of the guidelines and the fact that they address different types of users and situations. They are all designed to be essential to accessibility. Lastly, doing a general prioritization may not be the best strategy. In different projects, different things will be important. Doing an individual prioritization may be more useful.

It may be smart to estimate time needed to fulfil each guideline because the web development process is often deadline-oriented. If something can be done quickly, or time spent can be predicted, it is more likely to be taken into consideration. On the other hand, this needs an extensive study to be achieved. Time spent on the guidelines would have to be logged many times. It is also highly likely that time spent will depend on type of project and level of accessibility knowledge of team members. It could be argued that it is better not to estimate time, rather than estimating wrongly. Lastly, time estimation might result in favoring the guidelines that can be rapidly applied. Nevertheless, it would

be interesting looking further into these issues, as they were much sought after by the participants in this study, and may improve the quality and usability of the guidelines.

Adjust level of detail

The *level of detail* in how each guideline is explained, differs between the guidelines in the first version. Some are relatively short, while others are described in more detail. It was suggested to adjust this, to make all guidelines have approximately the same level of detail. The argument was the guidelines described in more detail, would seem more important than the ones with shorter descriptions, which are generally not the case. Our response to this suggestion, is that it is very challenging, if not impossible, to balance the level of detail of all guidelines so that they match exactly. This is because some issues need more detailed explanations to be comprehensible, while other topics can be described in very few words without loosing meaning. As an example, explaining how to bring accessibility into the web development process is a more diffuse issue than how to technically ensure accessibility. It will therefore be a natural consequence that the process-oriented guidelines are less specific and detailed. It could further be argued that some guidelines serve more as reminders to web developers rather than giving new information about accessibility. It would not be necessary to explain to a front-end developer how to use HTML5, or an interaction designer what good design practices are. It is natural that the guidelines focus on the aspects of accessibility that are outside of web developers' general competence area: WAI-ARIA, testing for accessibility, progressive enhancement and unobtrusive JavaScript. This might give a sense of building on existing knowledge when working with accessibility, and it highlights how accessibility in fact is a part of usability.

Guidelines that are extensively explained can make it easier for a developer to understand exactly what needs to be done, and how to technically achieve it. If so, a high level of detail could make the guidelines more usable. Many details can also make the guidelines too long and comprehensive. Too much detail in the technology-oriented guidelines might make them less transferable to future projects, or even ongoing projects not using the technology addressed in the guidelines. Further, a very detailed process description may not be useful or even possible, as every process is different.

Include other areas of web development

It was pointed out that there is little in the guidelines about interaction design, graphic design and information architecture. While these areas are certainly relevant for achieving accessibility, they are not in the scope of this project. It may have caused some confusion that a guideline called *Follow existing principles for good design* has been included. This guideline could be removed altogether because it is outside the scope. However, it has been kept for two reasons. Firstly, reminds developers that much of their existing competence is relevant for accessibility. Secondly, it is a reminder for future development to include more about these areas in coming versions. For now, there is a set of technology oriented guidelines and process oriented guidelines. In the future, design-oriented guidelines may also be included.

Make a check-list offering more information if needed

There was a general agreement that the most pedagogical way of presenting the guidelines is as a check-list which can be expanded to provide further information, also including

links to information from other sources. Bullet points should present the most important aspects. The guidelines would then function on two levels. The check-list would function as a tool to make sure everything is remembered and the guide would offer in depth guidance on each point. It was thought that this increases usability because developers can efficiently go through the guidelines and look further into the guidelines that are relevant for them.

6 Conclusions and further work

This chapter reflects on the findings, discusses the importance of this work and the next steps in this process.

Reflection on findings

This study has uncovered many RIA accessibility issues, but just as many suggestions for solutions. These solutions has been processed into guidelines aimed at creating more accessible web applications. The guidelines have been evaluated by web developers who have commented on both strengths and weaknesses.

The most prominent suggestions for changes to the guidelines, like the requests for time estimates and prioritized lists, makes us wonder if some of the most important issues are related to attitudes towards universal design in the current IT industry. Time is always of essence in a development process, but it may seem like there is a common attitude that universal design is too expensive and time-consuming, although a lot simply has to do with knowledge and training of the developers. We speculate if there is a general resistance against investing time and money in learning the skills of developing accessible IT solutions, but hope that the industry soon will see the benefits of acquiring this knowledge.

Contribution to the research field

To our knowledge, this is the first attempt to provide a set of accessibility guidelines focusing solely on front-end development with JavaScript, HTML and CSS. Other accessibility guidelines, such as WCAG and ATAG have different and larger focus areas. This contribution is important, because it focuses on the most complex areas of web development and accessibility; maintaining accessibility in highly dynamic and interactive applications.

The literature review performed as a part of this project, shows that there already is quite a lot of knowledge about the technical aspects. It is therefore the authors' belief that the most important contribution to the web accessibility research field will be related to the *process-oriented* guidelines.

Other fields within web development like programming, graphics, interaction design and information architecture are well established and well integrated in the web development process. Accessibility does not have that traction yet. It seems there are uncertainties as to how to integrate it into the process. This study has strong indications for a need for process-oriented accessibility guidelines. All participants genuinely appreciated this and mentioned they had not seen anything like that, including the accessibility expert. This project has also started the work of creating a set of process-oriented accessibility guidelines and has made several suggestions for continued work with these guidelines.It is highly likely that making the process of accessibility smooth and easy will solve many of the social issues with web accessibility, remove uncertainties

and make accessibility a natural part of the web development process in line with usability and other areas of development.

Future work

Although a lot of work was done to update the guidelines after the evaluation, not all the suggestions for changes were managed before finishing this project. Some things that are left should be done before the next round of interviews, for example providing a lengthier explanation of why each guideline is good for accessibility and the consequences of following it. Some suggestions should be further studied before making a decision of whether or not to place them in the guidelines, for example the extent of the issues with <article> and <section> and use of JavaScript to hide content and if or how it benefits accessibility. This should be studied before the next round of interviews.

Some issues that have emerged during the evaluation of these guidelines require extensive further studies across several iterations of guideline development. These are reliability of the guidelines, prioritization and time estimation of guidelines, guidelines related to other aspects of web development and further development of the process-oriented guidelines. It is the authors' believe that it is particularly important to continue studies and development of the process oriented guidelines.

References

[1] Juliana Cristina Braga, Rafael Jeferson Pezzuto Damaceno, Rodrigo Torres Leme, and Silvia Dotta. Accessibility study of rich web interface components.

[2] Ramón Voces Merayo. Rich internet applications (ria) y accesibilidad web. *Hipertext. net*, (9):2, 2011.

[3] Nádia Fernandes, Ana Sofia Batista, Daniel Costa, Carlos Duarte, and Luís Carriço. Three web accessibility evaluation perspectives for ria. In *Proceedings of the 10th International Cross-Disciplinary Conference on Web Accessibility*, page 12. ACM, 2013.

[4] *JavaScript usage statistics*, 2014 (accessed March 17, 2014). `http://trends.builtwith.com/docinfo/Javascript`.

[5] Alex Nederlof, Ali Mesbah, and Arie van Deursen. Software engineering for the web: the state of the practice. In *Companion Proceedings of the 36th International Conference on Software Engineering*, pages 4–13. ACM, 2014.

[6] Andy Brown, Caroline Jay, Alex Q Chen, and Simon Harper. The uptake of web 2.0 technologies, and its impact on visually disabled users. *Universal Access in the Information Society*, 11(2):185–199, 2012.

[7] Francisco J Garcia-Izquierdo and Raul Izquierdo. Is the browser the side for templating? *Internet Computing, IEEE*, 16(1):61–68, 2012.

[8] Lourdes Moreno, Paloma Martínez, Belen Ruiz, and Ana Iglesias. Toward an equal opportunity web: applications, standards, and tools that increase accessibility. *Computer*, 44(5):18–26, 2011.

[9] Joshue O Connor. *Pro HTML5 accessibility*. Apress, 2012.

[10] Walter Kern. Web 2.0-end of accessibility? analysis of most common problems with web 2.0 based applications regarding web accessibility. *International Journal of Public Information Systems*, 4(2), 2008.

[11] WHO. *World report on disabilities*, 2011. `http://whqlibdoc.who.int/publications/2011/9789240685215_eng.pdf?ua=1`.

[12] Giorgio Brajnik. Validity and reliability of web accessibility guidelines. In *Proceedings of the 11th international ACM SIGACCESS conference on Computers and accessibility*, pages 131–138. ACM, 2009.

[13] Jennifer Mankoff, Holly Fait, and Tu Tran. Is your web page accessible?: a comparative study of methods for assessing web page accessibility for the blind. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 41–50. ACM, 2005.

[14] Shari Trewin, Brian Cragun, Cal Swart, Jonathan Brezin, and John Richards. Accessibility challenges and tool features: an ibm web developer perspective. In *Proceedings of the 2010 international cross disciplinary conference on web accessibility (W4A)*, page 32. ACM, 2010.

[15] Yeliz Yesilada, Giorgio Brajnik, and Simon Harper. How much does expertise matter?: a barrier walkthrough study with experts and non-experts. In *Proceedings of the 11th international ACM SIGACCESS conference on Computers and accessibility*, pages 203–210. ACM, 2009.

[16] Alan Foley and Bob Regan. Web design for accessibility: Policies and practice. *AACE Journal*, 10(1):62–80, 2002.

[17] Steve Krug. *Don't make me think: A common sense approach to web usability*. Pearson Education India, 2005.

[18] Leonelo Dell Anhol Almeida and Maria Cecília Calani Baranauskas. Accessibility in rich internet applications: people and research. In *Proceedings of the 11th Brazilian Symposium on Human Factors in Computing Systems*, pages 3–12. Brazilian Computer Society, 2012.

[19] Eun-Ju Park, Yang-Won Lim, and Han-Kyu Lim. A study of web accessibility of websites built in html5-focusing on the top 100 most visited websites. *International Journal of Multimedia and Ubiquitous Engineering*, 9(4):247–256, 2014.

[20] B Lawson and S Faulkner. Html5 and accessibility. *MSDN Magazine*, 2011.

[21] Tim Wright. *Learning JavaScript: A Hands-On Guide to the Fundamentals of Modern JavaScript*. Addison-Wesley, 2012.

[22] Russ Ferguson and Christian Heilmann. *Beginning JavaScript with DOM Scripting and Ajax: Second Editon*. Apress, 2013.

[23] Eduardo Hideki Tanaka and Heloísa Vieira Da Rocha. Evaluation of web accessibility tools. In *Proceedings of the 10th Brazilian Symposium on on Human Factors in Computing Systems and the 5th Latin American Conference on Human-Computer Interaction*, pages 272–279. Brazilian Computer Society, 2011.