Improving Web Accessibility for Visually Impaired with Open Source Browser Extension

Sittiporn Auchariyabut and Yachai Limpiyakorn,
Department of Computer Engineering, Chulalongkorn University, Bangkok10330, Thailand
Sittiporn.a@student.chula.ac.th, Yachai.L@chula.ac.th

Abstract

Web accessibility means that people with disabilities can use the web. Web accessibility also benefits others including older people, and people in the rural areas. Currently, most websites have accessibility barriers that make it difficult or impossible for many people with disabilities to use the web. The Web Accessibility Initiative (WAI) was launched as an effort to improve the accessibility of the World Wide Web for people with disabilities. One of the roles of WAI is to develop guidelines and techniques that describe accessibility solutions for web software and web developers. A website optimized for accessibility will benefit everyone and worth investment as it gains a distinct competitive advantage and adds real value to the global community. This article presents the implementation of Firefox browser extension to adjust the HTML pages of existing web applications based on the recommendation of Web Content Accessibility Guidelines (WCAG) 2.0. The HTML adjustment is processed with two major steps, starting with the evaluation of HTML tags against the checkpoints defined as XPath 2.0 rules. In case the HTML element is nonconformance to WCAG 2.0, its location will then be returned to JavaScript for adjustments using the fixing rule satisfying the WCAG 2.0 success criteria. Examples of web accessibility adjustments are demonstrated. The browser extension is assessed compared to a web accessibility evaluation tool, for the correctness of identifying accessibility barriers.

Keywords: web accessibility, assistive technology, visually impaired, WCAG

1. Introduction

World Wide Web and Internet emerge as one of the greatest events of modern history. Most people today can hardly conceive of life without the internet. The web is an increasingly important resource in many aspects of life including education, commerce, government, healthcare, employment, news, entertainment, social inclusion, and more. With the click of a mouse, the world can be "at your fingertips". If you do not have a disability of any kind, you can browse around in the contents of the internet at any hour of days or nights.

However, many websites are currently developed with accessibility barriers that make it difficult or impossible for some people to use them, namely people with disabilities, people with temporary disabilities, people with changing abilities due to aging, and people in rural areas. As a basic human right, it is essential that the Web be accessible in order to provide equal access and equal opportunity to people with diverse abilities.

Referring to World Health Organization (WHO), there are 285 million visually impaired people with 39 million amount to blinds and 246 million classified as low vision people [1]. According to the findings reported in [2], out of 30 million webpages tested with WCAG 2.0 web accessibility testing standard, there are less than 4% of samples passing the success criteria. The result reflects that the accessibility of the internet to people with disabilities is still largely unrealized.
The web offers unprecedented opportunities, but how people with disabilities use the web. In 1997, the World Wide Web Consortium (W3C) launched the Web Accessibility Initiative (WAI) as an effort to improve the accessibility of the World Wide Web for people with disabilities. W3C also published Web Content Accessibility Guidelines (WCAG) as recommendations for making web contents more accessible. Following these guidelines will make web content more usable and accessible to a wider range of people including those with disabilities such as blindness and low vision.

It is untrue that the cost of developing accessible websites is more expensive and time-consuming compared to inaccessible ones. The benefits of providing access to a larger population almost always outweigh the time required by a knowledgeable developer to implement that accessibility. Carter and Markel [3] gave a list of reasons stating the convenience of developing accessible web applications. However, in case of existing websites, it would be impractical to re-implement them in order to satisfy greater accessibility of the web content to everyone. This research work therefore develops the browser extension for web accessibility adjustments of the existing websites. The accessibility adjustments focus on assisting people with visual disabilities.

The remainder of this paper is organized as follows: Section 2 describes web accessibility and introduces the readers with some relevant background knowledge. Section 3 describes the method and implementation details of the browser extension to improve the web accessibility for visually impaired. Section 4 provides examples of web accessibility adjustments and the evaluation against the WCAG 2.0 success criteria. Finally, Section 5 summarizes the work presented in this article and future direction of this research.

2. Background


In the HTML DOM, everything is a node (Figure 1): document itself is a document node, all HTML elements are element nodes, all HTML attributes are attribute nodes, and Text inside HTML elements are text nodes. When an HTML document is loaded into a web browser, it becomes a document object, which enables the access of data and adjustments of its structure.

2.2. Web Accessibility

2.2.1. Web Content Accessibility Guidelines (WCAG) [5]: Web Content Accessibility Guidelines is established by W3C. It provides the requirements for making web-based systems accessible. The recent WCAG 2.0 consists of several layers of guidance, including Principles, Guidelines, Success criteria, and Sufficient and Advisory techniques. At the top are four principles that provide the foundation for Web accessibility: Perceivable, Operable, Understandable, and Robust. Under the principles are guidelines, and there are 12 guidelines, for each of which, testable success criteria are provided. Three levels of conformance are defined: A (lowest), AA, AAA (highest).
2.2.2 Web Accessibility Initiative (WAI): The W3C Web Accessibility Initiative brings together people around the world from industry, disability organizations, government, and research labs to develop guidelines and resource to help make the web accessible to people with disabilities. WAI’s coverage of web accessibility includes web content (websites and web applications), authoring tools (such as content management systems (CMS) and blog software), browsers and other user agents, and W3C technical specifications, including WAI-ARIA [6] for Accessible Rich Internet Applications.

WAI-ARIA is an API between HTML and assistive technology as illustrated in Figure 2. To increase more web accessibility, WAI-ARIA allows (parts of) web pages to declare themselves as applications rather than as static documents, by adding role, state, and property information to dynamic web applications. As a result, the assistive technology will be triggered once the monitored tags have been updated.

People with disabilities sometimes use software, called assistive technologies, to interact with the web. Assistive technology accessibility includes (but is not limited to):

- Appropriate alternative text
- Form labels
- Table headers
- Logical heading structure
- Links that make sense out of context
- A logical, intuitive reading and navigation order
- Full keyboard accessibility
- Captions and transcripts

![Image: Figure 2. Architecture of WAI-ARIA Accessibility API](image)

2.3. WCAG Formalization with W3C Standards [7]

Web accessibility consists of a set of checkpoints which are rather expensive to spot or to evaluate the conformance with WCAG. The authors thus presents a W3C formalized rule-set version for automatable checkpoints according to WCAG 1.0. The WCAG checkpoints in this work were formalized as XPath 1.0 rules. However, no fixing is performed on the webpages for more accessibility.

2.4. Optical Character Recognition (OCR)

OCR is a technology that enables the conversion of different types of documents, such as scanned paper documents, PDF files, or images captured by digital cameras into editable and searchable data. It is a common method of digitizing printed texts so that they can
be electronically edited, searched, stored more compactly, displayed online, and used in machine processes such as machine translation, and text-to-speech.

Tesseract [8] is probably the most accurate open source OCR engine available. Combined with the Leptonica Image Processing Library, it can read a wide variety of image formats and convert them to text in over 60 languages. It was one of the top 3 engines in the 1995 UNLV Accuracy test. Tesseract is released under the Apache License 2.0, and it has been improved extensively by Google.

3. Implementation

In this work, the browser extension is implemented as the Firefox add-on. The method of accessibility adjustments includes the task to define and develop: 1) Checkpoint rules, and 2) Fixing rules. The checkpoints are formalized as XPath 2.0 rules, and a set of fixing rules is defined in the JavaScript. Figure 3 illustrates the WCAG 2.0 layers of guidance augmented with checkpoint rules and fixing rules defined in this research.

The fixing rules can be mainly divided into 2 categories: 1) objectively automatable rules, and 2) manual rules which require human judgment. In this work, the objectively automatable rules are defined and developed for HTML adjustments. However, the complex accessibility adjustments require knowledgeable human evaluation and involvement. The scope of accessibility adjustments covered by the current created browser extension is listed in Table 1. The adjustments are performed to meet WCAG 2.0 success criteria.

![Diagram](image)

**Figure 3. WCAG 2.0 Layers of Guidance Augmented with Rules defined in this Research**

**Table 1. Selected WCAG 2.0 Success Criteria**

<table>
<thead>
<tr>
<th>No.</th>
<th>Success Criteria</th>
<th>How To Meet</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1.1</td>
<td>Non-text Content</td>
<td>All non-text content that is presented to the user has a text alternative that serves the equivalent purpose.</td>
</tr>
<tr>
<td>1.3.1</td>
<td>Information and Relationships</td>
<td>Information, structure, and relationships conveyed through presentation can be programatically determined or are available in text.</td>
</tr>
<tr>
<td>1.3.2</td>
<td>Meaningful Sequence</td>
<td>When the sequence in which content is presented affects its meaning, a correct reading sequence can be programatically determined.</td>
</tr>
</tbody>
</table>
The web browser extension developed in this work is designed to enhance the web accessibility on Mozilla Firefox. Once it has been installed onto the web browser, the process of HTML adjustments is executed as described in the following 5 steps:

Step 1: Web browser requests data from the server by means of XMLHttpRequest (XHR).

Step 2: HTTP Interception resulting in sending the HTML via XHR to the web browser extension for working in step 3.

Step 3: Web browser extension performs the HTML adjustments using JavaScript based on the checkpoint validation with XPath 2.0 (Figure 4).

Step 4: Firefox web browser receives the adjusted HTML DOM structure from step 3.

Step 5: Web browser loads and displays the HTML displaying data such as CSS, images, media etc.

The web browser extension developed in this work is designed to enhance the web accessibility on Mozilla Firefox. Once it has been installed onto the web browser, the process of HTML adjustments is executed as described in the following 5 steps:

Step 1: Web browser requests data from the server by means of XMLHttpRequest (XHR).

Step 2: HTTP Interception resulting in sending the HTML via XHR to the web browser extension for working in step 3.

Step 3: Web browser extension performs the HTML adjustments using JavaScript based on the checkpoint validation with XPath 2.0 (Figure 4).

Step 4: Firefox web browser receives the adjusted HTML DOM structure from step 3.

Step 5: Web browser loads and displays the HTML displaying data such as CSS, images, media etc.

The web browser extension developed in this work is designed to enhance the web accessibility on Mozilla Firefox. Once it has been installed onto the web browser, the process of HTML adjustments is executed as described in the following 5 steps:

Step 1: Web browser requests data from the server by means of XMLHttpRequest (XHR).

Step 2: HTTP Interception resulting in sending the HTML via XHR to the web browser extension for working in step 3.

Step 3: Web browser extension performs the HTML adjustments using JavaScript based on the checkpoint validation with XPath 2.0 (Figure 4).

Step 4: Firefox web browser receives the adjusted HTML DOM structure from step 3.

Step 5: Web browser loads and displays the HTML displaying data such as CSS, images, media etc.

The web browser extension developed in this work is designed to enhance the web accessibility on Mozilla Firefox. Once it has been installed onto the web browser, the process of HTML adjustments is executed as described in the following 5 steps:

Step 1: Web browser requests data from the server by means of XMLHttpRequest (XHR).

Step 2: HTTP Interception resulting in sending the HTML via XHR to the web browser extension for working in step 3.

Step 3: Web browser extension performs the HTML adjustments using JavaScript based on the checkpoint validation with XPath 2.0 (Figure 4).

Step 4: Firefox web browser receives the adjusted HTML DOM structure from step 3.

Step 5: Web browser loads and displays the HTML displaying data such as CSS, images, media etc.

The web browser extension developed in this work is designed to enhance the web accessibility on Mozilla Firefox. Once it has been installed onto the web browser, the process of HTML adjustments is executed as described in the following 5 steps:

Step 1: Web browser requests data from the server by means of XMLHttpRequest (XHR).

Step 2: HTTP Interception resulting in sending the HTML via XHR to the web browser extension for working in step 3.

Step 3: Web browser extension performs the HTML adjustments using JavaScript based on the checkpoint validation with XPath 2.0 (Figure 4).

Step 4: Firefox web browser receives the adjusted HTML DOM structure from step 3.

Step 5: Web browser loads and displays the HTML displaying data such as CSS, images, media etc.

The web browser extension developed in this work is designed to enhance the web accessibility on Mozilla Firefox. Once it has been installed onto the web browser, the process of HTML adjustments is executed as described in the following 5 steps:

Step 1: Web browser requests data from the server by means of XMLHttpRequest (XHR).

Step 2: HTTP Interception resulting in sending the HTML via XHR to the web browser extension for working in step 3.

Step 3: Web browser extension performs the HTML adjustments using JavaScript based on the checkpoint validation with XPath 2.0 (Figure 4).

Step 4: Firefox web browser receives the adjusted HTML DOM structure from step 3.

Step 5: Web browser loads and displays the HTML displaying data such as CSS, images, media etc.

The web browser extension developed in this work is designed to enhance the web accessibility on Mozilla Firefox. Once it has been installed onto the web browser, the process of HTML adjustments is executed as described in the following 5 steps:

Step 1: Web browser requests data from the server by means of XMLHttpRequest (XHR).

Step 2: HTTP Interception resulting in sending the HTML via XHR to the web browser extension for working in step 3.

Step 3: Web browser extension performs the HTML adjustments using JavaScript based on the checkpoint validation with XPath 2.0 (Figure 4).

Step 4: Firefox web browser receives the adjusted HTML DOM structure from step 3.

Step 5: Web browser loads and displays the HTML displaying data such as CSS, images, media etc.

The web browser extension developed in this work is designed to enhance the web accessibility on Mozilla Firefox. Once it has been installed onto the web browser, the process of HTML adjustments is executed as described in the following 5 steps:

Step 1: Web browser requests data from the server by means of XMLHttpRequest (XHR).

Step 2: HTTP Interception resulting in sending the HTML via XHR to the web browser extension for working in step 3.

Step 3: Web browser extension performs the HTML adjustments using JavaScript based on the checkpoint validation with XPath 2.0 (Figure 4).

Step 4: Firefox web browser receives the adjusted HTML DOM structure from step 3.

Step 5: Web browser loads and displays the HTML displaying data such as CSS, images, media etc.

The web browser extension developed in this work is designed to enhance the web accessibility on Mozilla Firefox. Once it has been installed onto the web browser, the process of HTML adjustments is executed as described in the following 5 steps:

Step 1: Web browser requests data from the server by means of XMLHttpRequest (XHR).

Step 2: HTTP Interception resulting in sending the HTML via XHR to the web browser extension for working in step 3.

Step 3: Web browser extension performs the HTML adjustments using JavaScript based on the checkpoint validation with XPath 2.0 (Figure 4).

Step 4: Firefox web browser receives the adjusted HTML DOM structure from step 3.

Step 5: Web browser loads and displays the HTML displaying data such as CSS, images, media etc.
XPath is used to navigate through elements and attributes in an XML document. In this work, XPath 2.0 is used for the evaluation of HTML tags against the checkpoints defined as XPath rules. In case the HTML element is nonconformance to WCAG2.0, its location will be returned to JavaScript for adjustments using the fixing rule satisfying the associated success criteria.

Example of the XPath rule specifying the checkpoint based on the success criteria 1.4.7 “Low or No Background Audio” in Table 1 is shown as follow. This guideline recommends that the website should not contain background audio since this feature will interfere with the screen reader.

//audio[@autoplay] || //video[@autoplay]

Another example of the XPath rule based on the success criteria 2.4.5 “Multiple Ways” from Table 1 is shown as follow. This guideline recommends that the website should provide alternative means of accessing the web content. For example, the website should create the widget of table of content to inform the sequence of web headings so that the user would not scroll all over the webpage when looking for the desired content.

//h1/following-sibling::h2 //h2/following-sibling::h3

Example of JavaScript containing XPath and fixing rules 1.4.7 and 2.4.5 according to the success criteria 1.4.7 is illustrated in Figure 5.

```javascript
var audioautoplay = document.evaluate("//audio[@autoplay]",
document,
null,
XPathResult.UNORDERED_NODE_ITERATOR_TYPE,
null);
var thisNode = audioautoplay.iterateNext();
while (thisNode) {
    $(thisNode).removeAttr("autoplay");
}

var createTOC = document.evaluate("//h1//following-sibling::h2 //h2//following-sibling::h3",
document,
null,
XPathResult.UNORDERED_NODE_ITERATOR_TYPE,
null);
```

**Figure 5. Example XPath Defined to Locate HTML Tags for Adjustments**

The success criteria 1.1.1 “Non-text Content” recommends that the websites should provide text alternatives for all non-text content. For example, images should include equivalent alternative text in the markup/code, otherwise the image information is inaccessible for people who cannot see and use a screen reader that reads aloud the information on a page, including the alternative text for the visual image. When equivalent alternative text is provided, the information is available for everyone, including those who are visually impaired, people who turn off images on their mobile phone to lower bandwidth charges, people in rural areas with low bandwidth who turn off images to speed download, and others. The information is also available to technologies that cannot recognize images, such as search engines. Example of the XPath rule specifying the checkpoint for the adjustment to meet the success criteria 1.1.1 is defined as follow.

//img[@alt=''] //img[not(@alt)]

Figure 6 illustrates the solution how to provide alternative text for images with OCR technology. Tesseract [8] is used as the tool for image recognition. The browser extension starts with checking the tag against the Xpath rule, and then sends the image’s URL to the OCR server, which will then automatically convert the image to text, and return the result back to the webpage. Currently, the implemented Firefox extension can merely handle the case of image containing only text.
When assistive tools cannot recognize any change of tag inside the webpages that use Ajax, the property of WAI-ARIA, live region, will be applied to handle the situation. The MutationObserver is used to track webpage changing, and then the attribute live region will be inserted into that element as shown below.

```html
<div id="foo" aria-live="polite">
</div>
```

The architecture of the browser extension module is currently designed as shown in Figure 7. The browser extension module (.xpi) is developed using the Firefox builder. It is composed of two major components: 1) Add-on script, and 2) Content script. The Add-on script contains Add-on APIs, and Add-on code, that is the main process responsible for the construction of user interfaces and communication with the servers, OCR server. The Add-on script cannot directly interact with the webpages. It is the Content script that manages the webpages via the defined Xpath checkpoints, associated with the Fixing rules implemented in Javascript.

### 4. Demonstration and Evaluation

This section provides some examples of accessibility adjustments performed by the implemented Firefox Add-on. The web accessibility evaluation tool, AChecker [9], is used to examine the HTML pages for the conformance with accessibility standards—WCAG 2.0. AChecker can be used to self-evaluate the accessibility of single webpages to ensure that they meet the requirements of various international accessibility standards. The assessment of the implemented browser extension is conducted compared to the inspection result of AChecker.
4.1. Example 1

Figure 8 shows the initial HTML structure, of which the right hand side is the display corresponding to the left hand side HTML code. The initial HTML page has automatically play background sound function and it does not provide alternative means of web content access. The console dialogue in Figure 9 displays the evaluation result from executing the XPath rule that discovers the noncompliance HTML tags.

4.2. Example 2

Figure 11 illustrates the initial HTML structure, of which the right hand side is the display corresponding to the left hand side HTML code. The initial HTML page does not satisfy the success criteria 1.1.1 “Non-text Content” since there is no ‘alt’ attribute in <img> tag as a choice for assistive screen reader.
The browser extension starts with checking the tag against the XPath rule. Then the ID value and the URL of image are passed to the OCR server for image recognition. The browser console monitoring the sending and received messages is shown in Figure 12.

The output text returned from the OCR server is used to modify the tag element, i.e., inserting the attribute ‘alt’ and its value with the returned text as shown in Figure 13.

AChecker [9] is an open source web accessibility evaluation tool developed by the Inclusive Design Institute. AChecker is implemented as a web service, which will help identify accessibility barriers in HTML that may prevent people with disabilities from accessing to the web.

The webpage shown in Figure 14 was used as input to the AChecker for the accessibility inspection. The result is displayed in Figure 15. The two accessibility barriers are reported, consisting of 1) the noncompliance of the success criteria 1.1.1 “Non-text Content”, and 2) that of the success criteria 1.4.4 “Resize text”.

---

**Figure 11. HTML Structure and Display before Adjustment**

**Figure 12. Monitor of Communication with OCR Server**

**Figure 13. HTML Structure after Adjustment**

**Figure 14. Input Webpage for AChecker**

**Figure 15. Result of AChecker Inspection**
The same input webpage was examined by the implemented browser extension. The result is displayed in Figure 16 with the two identical barriers reported from AChecker. The webpage failed the criteria 1.1.1 since there does not exist the alternative text for the assistive screen reader. While the criteria 1.4.4 is not satisfied because the assistive screen readers cannot understand what element ‘bold’ means. Next, the adjustment was then performed. To satisfy the success criteria 1.1.1, the method using OCR technology as described earlier is applied to fix the barrier. To meet the success criteria 1.4.4, the element ‘bold’ is changed to the element ‘strong’ so that screen readers can understand and read text in a more emphasized tone.

![Figure 14. Webpage as Input for Accessibility Inspection](image1)

![Figure 15. AChecker Report of Noncompliance to WCAG 2.0](image2)

Once the input webpage had been fixed, it was re-input to the AChecker for re-inspection. The result is displayed in Figure 17, reporting that the webpage passed the accessibility inspection in this round.
5. Conclusion

The web offers so many opportunities to people with disabilities that are unavailable through any other media. However, if a web site is not created with web accessibility principle, the web’s potential is falling short. Studies reported that developing accessible websites gains economical advantages, such as better search results, reduced maintenance cost, and increased audience reach. Making a website accessible can be easily implemented if they are planned from the beginning of web site development. However, fixing inaccessible websites would require significant effort, especially sites that were not originally coded properly with standard XHTML markup, including sites with certain types of content such as multimedia. This article presents a prototype of the Firefox browser extension implemented to improve the web accessibility for visually impaired. The prototype covers some adjustment performed to satisfy selected success criteria of WCAG 2.0, for example, the adjustment to provide alternative text for images. More enhancement can be further developed on the image processing, such as fixing text lines (deskew and dewarp text), fixing illumination of images, binarizing and de-noising images. Additionally, to ease the adjustment, CSS (Cascading Style Sheets) could be introduced to provide a better way to style HTML elements.

The implemented browser extension is assessed compared to the web accessibility evaluation tool, AChecker. Both of them provide the same inspection result of identifying the web accessibility barriers for people with visual disabilities. The browser extension also correctly adjusted the accessibility barriers to satisfy the WCAG 2.0 success criteria.

References


Author

Sittiporn Autchariyabut, he received his bachelor degree in Multimedia and Internet System Engineering from Bangkok University in 2010. Currently, he is pursuing the Master degree in Software Engineering at Department of Computer Engineering, Chulalongkorn University, Bangkok 10330, Thailand.