Comparative Analysis of Accessibility Testing Tools and Their Limitations in RIAs

Obianuju Okafor(✉), Wajdi Aljedaani, and Stephanie Ludi

Department of Computer Science and Engineering, University of North Texas, Denton, TX, USA
{obianujuokafor,wajdialjedaani,stephaieludi}@unt.edu

Abstract. Accessibility is a required quality for websites today, and several tools exist to test for this quality. These tools are highly advantageous, but sadly they also have some limitations. A particular set of challenges they face is in the evaluation of Rich Internet Applications (RIAs). In this paper, we carry out an experiment to compare and analyze different accessibility testing tools as they evaluate 10 educational websites. We judged these tools based on their error detection, guideline coverage, speed, similarity to one another, and their relative performance when evaluating RIAs. The experiment findings revealed the strength and limitations of each tool. The results of this experiment also exposed that there are many guidelines and success criteria that accessibility testing tools are not able to cover, and that some evaluation tools are similar to each other in terms of the results they produce. Lastly, this experiment highlights a discrepancy in the behavior of the tools when evaluating RIAs compared to when evaluating static websites, although some more than others. This experiment has some limitations which we presented. As a future work, we intend to work with an expert to determine the accuracy of the results produced from the experiment. We also intend to delve deeper into the limitations of these tools and come up with possible solutions.

Keywords: Accessibility · Accessibility evaluation tools · Online education · Web Content Accessibility Guideline

1 Introduction

`Web accessibility` is the quality of a web application or website that determines how it can be used by people with the widest possible range of capabilities [14]. The more accessible a website is, the more diverse the people who can use it are, including people with disabilities like blindness, deafness, etc. Providing equal access to online services is required by law in many countries, particularly their government, educational institute websites, and learning management systems [11]. Regardless of the law, it is important for every website to be accessible to everyone to avoid excluding anyone.

V. G. Duffy et al. (Eds.): HCII 2022, LNCS 13521, pp. 479–500, 2022.
https://doi.org/10.1007/978-3-031-17902-0_34

As beneficial as accessibility is, websites today are either partially or entirely inaccessible [13]. According to the 2020 Web Accessibility Annual Report, 98% of websites in the United States are not accessible [7]. This is could be due to a number of reasons, such as web developers not having sufficient knowledge concerning accessibility and how to achieve it in websites [18], and there not being an efficient way to test for accessibility compliance. For a website to be considered accessible, it has to meet the requirements of accessibility guidelines [14], such as the Web Content Accessibility Guidelines (WCAG)[1], Section 806[2], BITV[3], ADA[4], etc.

There are three ways to test websites for conformance to an accessibility guideline: manually, automatically, or combining both methods [16]. In manual evaluation, tests are carried out by humans using the guidelines as a rule book. The automated testing approach involves the use of software tools with no human intervention. The last method of testing combines the features of the other two, testing is done partly by software tools and partly by humans, e.g., SAMBA [4].

In this paper, we focus on the automatic approach of testing, which involves the use of software tools. This method of testing for accessibility has several advantages [8]. A particular benefit it has is that it is an easy and efficient method for developers who do not have expertise inaccessibility to ensure their websites comply with accessibility guidelines. This approach also has several limitations. One challenge they face is in evaluating Rich Internet Applications (RIAs) [29]. In this context, the goal of this paper is to compare and analyze several accessibility evaluation tools in terms of error detection, guideline coverage, speed, correctness, tool similarity, and their relative performance with respect to RIAs.

The paper makes the following contributions:

– This paper conducts a comparative analysis of four existing accessibility evaluation tools. We provide experimental results highlighting each tool's weaknesses and strengths.
– This paper also presents the limitations these tools have regarding testing for accessibility in RIAs.
– We expect that our findings will provide researchers and developers with an overview of which tools are suitable for the evaluation of static and dynamic websites. It will also help them get a better understanding of accessibility.
– Our findings also highlight the remaining opportunities and research directions for better design of tools that evaluate RIAs.

The rest of this paper is organized as follows. We present background information in Sect. 2. In Sect. 3, discusses the other research that has been carried out relating to this research topic. We explain the approach we took when carrying out the experiment in Sect. 4. In Sect. 5, we present the results of this experiment and discuss them. In Sect. 5.1 presents the limitation of our study. We conclude and discuss some future works in Sect. 6.

[1] https://www.w3.org/WAI/standards-guidelines/wcag/.
[2] https://www.section508.gov/.
[3] https://www.accessi.org/bitv.
[4] https://www.access-board.gov/ada/.

2 Background

This section presents background information on the research problem. It first gives an overview of the Web Content Accessibility Guidelines (WCAG), then it discusses the existing tools used to automatically check for compliance to these guidelines and highlights some of their limitations. Finally, it talks about RIAs and the challenges they pose to accessibility evaluation tools.

2.1 Web Content Accessibility Guidelines (WCAG)

`WCAG` was created by the World Wide Web Consortium (W3C)[5]. The goal for creating WCAG is to provide a single standard for the creation of accessible web content that meets the needs of individuals, organizations, and governments globally [28]. The WCAG standards are grouped into four principles: perceivable, operable, understandable, and robust. Under these four principles, there are a total of 12–13 guidelines, depending on the version. Each guideline contains testable success criteria, which are associated with different levels of conformance: A (lowest), AA, and AAA (highest). The success criteria are what determines conformance to WCAG. In Table 1 we present the four principles and their respective guidelines, along with the number of success criteria in each guideline.

There are several versions of WCAG, but in this paper, we will be focusing on versions WCAG 2.0[6] and WCAG 2.1[7]. WCAG 2.0 was published in December 2008 and adopted in October 2012 by the International Organization of Standardization (ISO) as ISO an standard [28]. WCAG 2.1 was published in June 2018 [28]. WCAG 2.0 has a total of 61 success criteria, while WCAG 2.1 has a total of 78 success criteria. WCAG 2.1 extends WCAG 2.0, therefore all success criteria from 2.0 are included in 2.1, but there are 17 additional success criteria in WCAG 2.1 that are not in WCAG 2.0. This means that a website that meets WCAG 2.1 also meets the requirements of WCAG 2.0 [28].

2.2 Accessibility Evaluation Tools

`Accessibility evaluation tools` are websites, web applications, or desktop applications that help developers determine if web contents meet accessibility guidelines [3]. These tools verify compliance by testing for each success criteria of a guideline. Figure 1 shows the website of the AChecker tool, one of the tools used in this experiment. Table 2 gives some examples of these tools and their features.

The use of evaluation tools has several benefits [8]. It is a fast and easy way to obtain information on the accessibility level of a website and it is affordable in terms of evaluating large numbers of web pages [23]. However, as beneficial

[5] https://www.w3.org/.
[6] https://www.w3.org/TR/WCAG20/.
[7] https://www.w3.org/TR/WCAG21/.

Table 1. Organization of WCAG 2.0 and WCAG 2.1

Principles	Guidelines	Number of success criteria	
		WCAG 2.0	WCAG 2.1
Perceivable	1.1 Text Alternatives	1	1
	1.2 Time-based Media	9	9
	1.3 Adaptable	3	6
	1.4. Distinguishable	9	13
Operable	2.1. Keyboard Accessible	3	4
	2.2. Enough Time	5	6
	2.3. Seizures and Physical Reactions	2	3
	2.4. Navigable	10	10
	2.5. Input Modalities	–	6
Understandable	3.1. Readable	6	6
	3.2. Predictable	5	5
	3.3. Input Assistance	6	6
Robust	4.1. Compatible	2	3
	Total	**61**	**78**

as these tools are, there are still drawbacks to using them. Apart from the fact that they are not able to check for all guidelines [21,22], and that they are not accurate, i.e., may produce false negatives or positives [1,23], they also face challenges when evaluating RIAs [29].

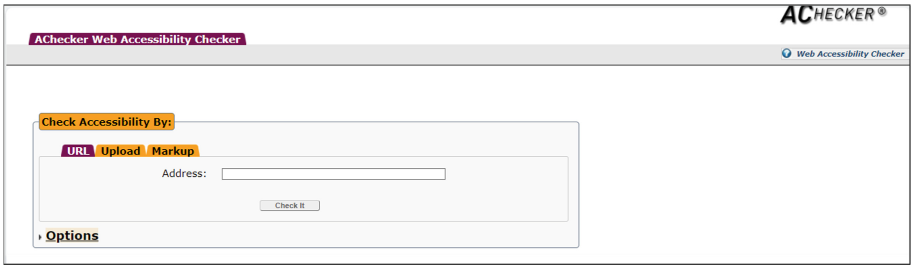


Fig. 1. Interface of the AChecker tool, via AChecker's website (https://achecker.achecks.ca/checker/index.php)

2.3 Rich Internet Applications (RIAs)

According to the author in [6], `Rich Internet Applications` are web applications or websites aimed to provide users with a similar desktop experience. RIAs are dynamic in nature and offer a richer and more interactive environment compared to traditional websites [29]. This is due to the use of a new set of technologies, such as DHTML, XML-HTTPRequest, DOM Events, etc. It was later discovered that these new technologies and capabilities introduce new sets of accessibility challenges to the Web [12]. Accessibility evaluation tools are not able to evaluate the dynamically generated contents that constitute RIAs [29]. These tools possess limited crawling capabilities and are only able to analyze static HTML content, therefore, DOM elements often go unnoticed [29].

To address the accessibility challenges posed by RIAs, the Web Accessibility Initiative (WAI) created a new specification called the Accessible Rich Internet Applications ARIA [31]. This specification stipulates how to make web pages, particularly dynamic content, more accessible [30]. Sadly, the majority of the existing accessibility evaluation tools are not able to evaluate websites using the ARIA specification.

3 Related Work

This section highlights several prior works that profoundly influenced our approach. We divided the related work into three sections: comparison of accessibility evaluation tools, accessibility evaluation for educational websites, and accessibility in Rich Internet Applications.

3.1 Comparing Accessibility Evaluation Tools

Previous studies have performed comparisons of accessibility evaluation tools. In the study by Christian et al. [20], conducted a comparative analysis of the performance of online accessibility evaluation tools. They compared six tools and found that the best tools were AChecker and TAW, with AChecker being the better tool of the two. They also proposed the classification of accessibility tools according to their usage and functionality. The research conducted in [1] compared five accessibility evaluation tools. In this study, faults were intentionally injected into web pages and then tested to investigate how the tools can detect injected faults. Their results showed that the tools do not discover all accessibility faults in web pages and that they produce inaccurate results.

Vigo et al. [24], performed an empirical investigation on six tools to show their capabilities in terms of coverage, completeness, and correctness regarding WCAG 2.0 conformance. They performed their investigation by using the semi-automated method of testing, and they combined the results of the tools LIFT and Bobby with human evaluation. They deduced that relying on automated tests alone has negative effects and could lead to unwanted consequences.

3.2 Accessibility Evaluation for Educational Websites

Several studies have conducted an experiment to evaluate educational websites. For example, Abdullah Alsaeedi [2] evaluated six Saudi universities' websites by using two tools, WAVE and Siteimprove. The author proposed a novel framework to compare the performance of the tools. Another study [9] analyzed 44 higher education websites in India. Their analysis was performed with two tools, TAW and aXe, and the results showed that TAw identified more issues than aXE. A more recent study [5] performed a systematic review of empirical studies on educational websites for web accessibility. The study identified 25 studies. The authors' recommendation is to improve the automatic evaluation tools.

3.3 Accessibility in Rich Internet Applications

In the third category are studies relating to RIAs and how to make them accessible. The first study we examined was by Watanabe et al. [29]. They describe an approach for testing accessibility requirements in RIAs that involves the use of acceptance tests. They also implemented a tool that automatically runs test cases and considers assistive technology user scenarios to raise accessible design flaws. Another study relating to RIAs was done by Linaje et al. [12]. They combined two methods from previous works, the RUX-Method and SAW, to provide accessibility features to Rich Internet Applications. To combine these two techniques, they used ontoRUX, an ontology based on WAI-ARIA. The result of this is a process that makes Web applications with RIA features accessible [12].

Our study is similar to all 3 categories of related work. We compare 4 accessibility tools to identify how they perform in terms of error detection, guideline coverage, speed, and similarity to one another, just like some of the studies in the first category. Furthermore, like the studies in the second category, we used the tools to evaluate computer science educational websites. Some of the websites we will be evaluating are RIAs, akin to the studies in the 3rd category. Our study differs from the others because we used the tools to evaluate both static websites and RIAs, and we have analyzed the tools' relative performance when evaluating the two types of websites.

4 Experiment

This experiment was carried out to compare web accessibility evaluation tools and analyze their performance in terms of error detection, guideline coverage, speed, tool similarity, and relative performance when evaluating RIAs.

In this study, we aim to address the following research questions:

- `RQ1`: Which tool is able to detect the most errors?
- `RQ2`: How is the coverage of the tools in terms of WCAG 2.0 or WCAG 2.1 guidelines?
- `RQ3`: How fast are the tools when evaluating websites for accessibility?
- `RQ4`: Which tools produce the most similar results?
- `RQ5`: What is the relative performance of the tools when evaluating Rich Internet Applications?

To achieve our research goals, we followed four main steps:

4.1 Tool Selection

We conducted research to determine what are the existing accessibility evaluation tools. We found several articles on some of the best tools being used to evaluate websites for accessibility [10,15,17,19,25]. We selected four of some of the most recommended tools. They are AChecker[8], FAE[9], TAW[10], and WAVE[11].

The tools which we chose have in common their ability to test web pages against the WCAG 2.0 or WCAG 2.1 guideline, they are free, web-based, and can evaluate websites using their URL. Some of the tools also evaluate websites using their source code, or by uploading their code files. Table 2 gives more information on each of the selected tools.

Table 2. Accessibility evaluation tools and their features

Tool	Guideline	License	Deployment	Report format
AChecker	BITV 1.0 (Level 2), Section 508, Stanca Act, WCAG 1.0 (A, AA, AAA), WCAG 2.0 (A, AA, AAA)	Free, Commercial	Website, Desktop Application	HTML
FAE	WCAG 2.1 (A, AA, AAA), ARIA	Free	Website	HTML, Email, CSV
TAW	WCAG 2.0 (A, AA, AAA)	Free	Website, Desktop Application	HTML, Email
WAVE	WCAG 2.1 (A, AA)	Free, Commercial	Website, Browser extension	HTML

4.2 Website Selection

In this experiment, we evaluate two categories of websites, static and dynamic websites (RIAs). In this context, static websites are websites in which the content of each page does not change, while dynamic websites are websites whose contents are generated dynamically.

We selected the websites in the context of computer science (CS) education. They are websites that can be used to learn different CS concepts. To find websites for the study, we did a Google search for websites that offer CS tutorials, and we found a total of 52 websites. We selected 10 out of 52, 5 static websites and 5 dynamic websites. We made sure to choose the websites that do not require a login, as that can affect the tools' ability to evaluate them. Table 3 shows the selected static websites and their URLs, and Table 4 shows the selected dynamic websites and their URLs.

[8] https://achecker.achecks.ca/checker/index.php.
[9] https://fae.disability.illinois.edu/.
[10] https://www.tawdis.net/resumen.
[11] https://wave.webaim.org/.

Table 3. Static websites and their URL

Website	URL
Geeksforgreek	https://www.geeksforgeeks.org/
w3schools	https://www.w3schools.com/
KD Nuggets	https://www.kdnuggets.com/tutorials/index.html
Guru99	https://www.guru99.com/
TutorialPoint	https://www.tutorialspoint.com/

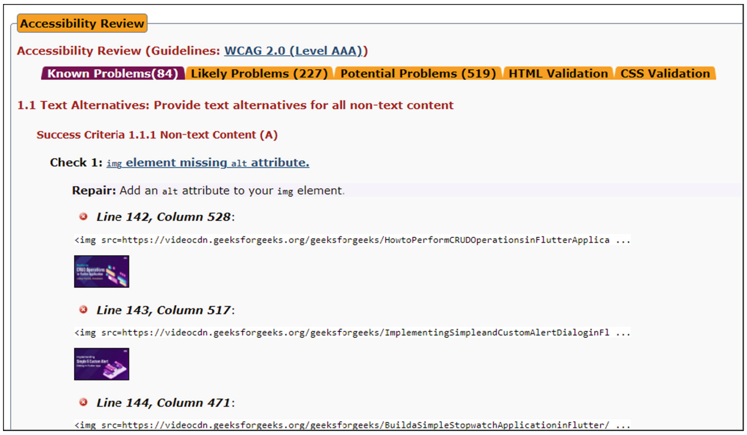
Table 4. Dynamic websites and their URL

Website	URL
Snap	https://snap.berkeley.edu/snapsource/snap.html
Blockly	https://blockly-demo.appspot.com/static/demos/code/index.html
Code.org	https://studio.code.org/s/dance-2019/lessons/1/levels/1
Vidcode	https://www.vidcode.com/project/intro
Make Code	https://makecode.microbit.org/#editor

4.3 Experimental Procedure

During the experiment, the 4 selected tools (see Table 2) were applied to the 10 selected websites(see Tables 3 and 4). Thus, for every website, we carried out 4 tests, leading to a total number of 40 tests that were carried out in this experiment. Only a single page of each website was evaluated, often times this was the home page.

For all tools except WAVE, we configured the settings to ensure that they were evaluating websites using the proper guideline and conformance level. In Achecker, we selected **WCAG 2.0** as the guideline, and level **AAA** as the conformance level. TAW did not have the option to select a guideline, but we were able to select conformance level **AAA**. FAE had the most options to choose from. We could choose the ruleset (guideline), depth of evaluation, number of pages to be evaluated, etc. We selected the guideline **HTML4 Legacy Technique**, which entails the WCAG 2.0 guideline levels A, AA, and AAA, as well as HTML4, HTML5, and ARIA techniques. We selected the depth as **Top-level page only**, and the number of pages as **5**. Since we selected **Top-level page only**, only a single page was evaluated. WAVE was the only tool that did not provide options to choose from. It was built to evaluate websites using WCAG 2.1 and conformance levels A and AA only.


Accessibility Review

Accessibility Review (Guidelines: WCAG 2.0 (Level AAA))

Known Problems(84) | Likely Problems (227) | Potential Problems (519) | HTML Validation | CSS Validation

1.1 Text Alternatives: Provide text alternatives for all non-text content

Success Criteria 1.1.1 Non-text Content (A)

Check 1: `img` element missing `alt` attribute.

Repair: Add an `alt` attribute to your `img` element.

Line 142, Column 528:

```
<img src=https://videocdn.geeksforgeeks.org/geeksforgeeks/HowtoPerformCRUDOperationsinFlutterApplica ...
```

Line 143, Column 517:

```
<img src=https://videocdn.geeksforgeeks.org/geeksforgeeks/ImplementingSimpleandCustomAlertDialoginFl ...
```

Line 144, Column 471:

```
<img src=https://videocdn.geeksforgeeks.org/geeksforgeeks/BuildaSimpleStopwatchApplicationinFlutter/ ...
```

Fig. 2. Evaluation report generated by the AChecker tool (https://achecker.achecks.ca/checker/index.php)

To evaluate a website, the URL of the selected website was entered into each tool. Before starting each evaluation, we started a timer, and once the evaluation was done, we stopped the timer and recorded the time taken. After each test, we were presented with an evaluation report showing the total number of errors found, error description, which success criteria failure caused the error, a suggestion on how to fix the error, other potential errors, etc. The 40 tests that were carried out in the experiment produced 40 evaluation reports.

We observed that all tools did not produce results in the same format. Some tools' evaluation report was displayed on the screen as HTML content (see Fig. 2), while the evaluation report was sent through email in some cases. Additionally, some tools reported their errors according to HTML elements on the screen, e.g., images, links, forms, etc., while some tools categorized the results according to the guidelines failed. Due to this, we had to extract the data from each report and organize it in a uniform format. We created an excel sheet. For each test conducted, we entered into the excel sheet, evaluation time, total number of errors found, error details, success criteria failed, as well as any comments we had. We analyzed the data and wrote down our findings. We present the tables, diagrams, results, and discussion in the next section.

5 Result and Discussion

In this section, we present the results of the experiment. We also further discuss these results. The result of this experiment is divided into five sections, with each section addressing the research questions RQ1, RQ2, RQ3, RQ4, and RQ5, respectively.

Table 5. The different types of errors detected and the tools that found them.

Error	WAVE	Achecker	FAE	TAW
Image missing alternative text	Y	Y	Y	Y
Input element is missing form label	Y	Y	Y	Y
Empty link	Y	Y	Y	Y
Empty heading	Y	N	N	Y
Empty button	Y	N	Y	N
Contrast errors	Y	Y	Y	N
Web page well-formedness	N	N	N	Y
b (bold) element used	N	Y	N	N
i (italic) element used	N	Y	N	N
Font used	N	Y	N	N
Incorrect Header nesting	N	Y	Y	Y
Two headers of the same level with no content in between	N	N	N	Y
No h1 element in the document	N	N	Y	Y
Data cells does not use headers or ID attribute	N	Y	Y	N
Data table with both row and column headers does not use scope to identify cell	N	Y	N	N
Data table elements used in layout tables	N	N	N	Y
id attribute is not unique	N	Y	N	N
Form with no standard submission method	N	N	Y	Y
Frames without title	N	N	N	Y
Links with same link text but different destinations	N	N	N	Y
Consecutive text and image links to the same resource	N	N	N	Y
The language of the document is not identified or is invalid	Y	Y	Y	Y
Elements with event handlers must have roles	N	N	Y	N
Presence of empty lists	N	N	N	Y
Use of device-dependent event handlers	N	N	Y	Y
ARIA values must be valid	N	N	Y	N
Broken ARIA menu	Y	N	N	N
Use of labels to modify the presentation	N	N	N	Y
Widget labels must be descriptive	N	N	Y	N
Widgets must have label	N	N	Y	N
Label must reference control	N	N	Y	N
Role must have parent	N	N	Y	N
iframe must have accessible name	N	N	Y	N
Selection form control without grouping	N	N	N	Y
“div” elements that simulate paragraphs	N	N	N	Y
Total	**8**	**12**	**18**	**19**

RQ1: Which tool is able to detect the most errors?
At the end of each website evaluation, an evaluation report was generated. Part of the things shown in this report is the total number of errors found and the error description (see Fig. 2). In this section, we present the different types of errors found by each tool and the total number of errors reported in each website per tool.

Result. After compiling the results, we observed that there are 35 types of errors that were detected by the tools. Table 5 is a list of errors found across all websites by all tools. The most commonly reported error was *‘Missing form label’*, it was found as an error in all 10 websites. The next most common error reported was *‘Image missing alternative text’* and *‘Empty link’*. They were both found in 70% of websites.

Table 6. Total number of errors detected per tool in each static website

Website	WAVE	AChecker	FAE	TAW
Geeksforgeeks	129	83	211	164
KDNuggets	8	54	7	35
W3School	26	48	–	91
Guru99	30	203	69	102
TutorialPoint	71	121	34	35
Average	52.8	101.8	80.25	85.4

Table 7. Total number of errors detected per tool in each dynamic website

Website	WAVE	AChecker	FAE	TAW
Snap	2	2	3	4
Code.org	26	14	8	68
MakeCode	2	0	2	2
Blockly	4	0	22	18
Vidcode	13	4	23	34
Average	9.4	4	11.6	25.2

No tool was able to detect all 35 errors. However, some tools detected more types of errors than others. TAW was able to detect the most type of errors. It found 19 different types of error out of 35, across all 10 websites. TAW could detect several errors that the others could not e.g. *'Web page well-formedness'*, *'Frames without title'*, etc. However, all tools except TAW were able to detect *'Contrast errors'*. TAW is closely followed by FAE, which was able to detect 18 types of errors. The next tool on the rank is Achecker. Although it could detect only 12 out of 35 errors, it is the only tool that reports text errors such as the *'Fonts used'*, *'i (italic) element used'*, and *'b (bold) element used'*. The tool that detected the least types of errors is WAVE. It detected just 8 out of 35 types of error. The only error it could detect that others could not was *'broken ARIA menu'*.

Regarding the number of errors found in each website, Tables 6 and 7 show the number of errors each tool detected in static and dynamic websites, respectively. The total number of errors reported is derived from the different types of errors found in a website multiplied by the number of elements that cause these errors. Some tools found more types of errors than the others, while for each type of error, some tools found more violating elements. All of this contributed to the difference in the number of errors reported by each tool.

Looking at Table 6, we can see that on average, Achecker reported the highest number of errors in static websites, with a mean of 101.8. It is followed by TAW, with a mean of 85.4. FAE had a mean of 80.25, making it the 3rd in this category. FAE might have had a higher mean, but it was not able to evaluate the website *'W3school'*. WAVE had the lowest average of 52.8, which is almost half the average of Achecker. In the dynamic website category (see Table 7), TAW had the highest mean of 25.2, while Achecker had the lowest mean of 4. The average of WAVE and FAE was 9.4 and 11.6, respectively.

Discussion. Although Achecker had the highest average for the number of errors in static websites, we can deduce that TAW performed the best in terms of error detection. It detected the most types of errors. It had the highest average number of errors in dynamic websites and the second highest average number of errors in static websites. TAW and Achecker's performance in terms of detecting a higher number of errors in static and dynamic websites than WAVE and FAE, was unexpected. TAW and Achecker both check against WCAG 2.0, which has fewer guidelines to violate than WCAG 2.1, which WAVE and FAE check against.

While going through the results of each tool, we observed that TAW and Achecker evaluation reports sometimes had errors that were duplicates. TAW and Achecker categorize errors by success criteria (see Fig. 2), hence, they often had errors that were repeated in the report, as one error could violate multiple success criteria. In contrast, WAVE and FAE grouped errors according to the elements that cause them, e.g., images, forms, etc., hence the errors were never repeated. Additionally, in the event that all tools detected a particular error in a website, TAW and Achecker often reported a higher number of violating elements than FAE and WAVE. All of these could be the reason Achecker and TAW had higher number of errors compared to WAVE and FAE. WAVE's low numbers could also be due to the fact that it could only check for conformance levels A and AA, while the other tools could check up to conformance level AAA.

As the tools often reported different numbers for the number of errors found in a website, it is hard to determine which tool produced the most accurate result. Some tools may have reported non-existing or trivial problems (false positives) or they might have missed true problems (false negatives). We will need to work with an accessibility expert to determine which tool has the most accurate results.

RQ2: How is the coverage of the tools in terms of WCAG 2.0 or WCAG 2.1 guidelines?
When evaluating a website, the tools check for adherence to the success criteria of the WCAG 2.0 guideline, in the case of Achecker and TAW, or the WCAG 2.1 guideline, in the case of WAVE and FAE. In this section, we analyze the tools in terms of their coverage of the WCAG 2.0 or WCAG 2.1 guideline.

Result. Table 8 shows which guidelines and success criteria failure at least one of the tools was able to detect across all websites. You can find the full list of guidelines and success criteria for WCAG 2.0 in [26], and for WCAG 2.1 in [27]. The green tick signifies (✓)that the tool was able to detect those success criteria in at least one of the 10 websites. If a tool never detected a success criteria failure in any website, a red (✕) is shown. Table 9 shows how many success criteria failures were detected in static and dynamic websites.

Looking at Table 8, we can see that there were only 9 out of 13 possible guidelines failures that were detected. The following guidelines were never covered by

Table 8. Table showing the coverage of the tools in terms of WCAG 2.0 and WCAG 2.1

Principle	Guideline	Success Criteria	WAVE	Achecker	FAE	TAW
Perceivable	**1.1**	1.1.1 Non-text Content (A)	✓	✓	✓	✓
	1.3	1.3.1 Info and Relationships (A)	✓	✓	✓	✓
		1.4.1 Use of Color (A)	✗	✗	✓	✗
		1.4.3 Contrast (Minimum) (AA)	✓	✓	✓	✗
	1.4	1.4.4 Resize text (AA)	✗	✓	✗	✗
		1.4.6 Contrast (Enhanced) (AAA)	✗	✓	✓	✗
Operable	**2.1**	2.1.1 Keyboard (A)	✓	✗	✗	✗
		2.1.3 Keyboard (No Exception) (AAA)	✗	✗	✗	✓
		2.4.1 Bypass Blocks (A)	✓	✗	✓	✗
		2.4.4 Link Purpose (A)	✓	✓	✓	✓
	2.4	2.4.6 Headings and Labels (AA)	✓	✓	✓	✗
		2.4.9-Link Purpose (Link Only) (AAA)	✗	✗	✓	✓
		2.4.10-Section Headings (AAA)	✗	✗	✓	✓
Understandable	**3.1**	3.1.1 Language of Page (A)	✓	✓	✓	✓
		3.2.2 On input (A)	✗	✗	✓	✓
	3.2	3.2.3 Consistent Navigation (AA)	✗	✗	✓	✗
		3.2.4 Consistent Identification (AA)	✗	✗	✓	✗
	3.3	3.3.2 Labels or Instructions (A)	✓	✓	✓	✓
		4.1.1-Parsing (A)	✓	✓	✗	✓
Robust	**4.0**	4.1.2 Name, Role, Value (A)	✗	✗	✓	✓
Total	9	20	10	10	16	11

Table 9. Total number of success criteria detected in static and dynamic websites

	WAVE	Achecker	FAE	TAW
Static websites	7	9	12	10
Dynamic websites	9	5	16	10

any of the tools: 1.2, 2.2, 2.3, 2.5. Because some of the tools check for WCAG 2.1, there are 78 possible success criteria that could be detected. However, only 20 success criteria was covered. All of the success criteria detected can be found in both WCAG 2.0 and WCAG 2.1 guidelines. Additionally, over half of the success criteria failures reported had conformance level **A**.

FAE had the highest overall guideline coverage and the highest coverage in both static and dynamic websites. It detected 12 success criteria failures in static websites, while it detected 16 success criteria failures in dynamic websites. It is followed by TAW, which detected 11 different success criteria failures in total. It detected 10 success criteria failures in both static and dynamic websites. Achecker and Wave were both able to detect 10 types of success criteria failures, however, WAVE detected a higher number of success criteria failures in dynamic websites compared to Achecker. It found 9 success criteria failures while Achecker found 5. Achecker reported a higher number of success criteria failures in static websites compared to WAVE. It detected 9 success criteria violations while WAVE detected 7.

Discussion. Given that TAW performed best in terms of error detection, one would expect that it will also have the highest coverage in terms of guidelines. However, that was not the case. FAE had the most coverage instead. FAE's performance could be due to the fact that it uses a combination of techniques and guidelines when evaluating websites, e.g., WCAG 2.1, ARIA, HTML4, and HTML5 specifications. In addition, it performed almost as well as TAW in terms of detecting the most type of errors. Its ability to detect different types of error and its use of several techniques, might have led to it being able to detect the most success criteria violations.

WAVE's low guideline coverage in static websites was expected, as it is the only tool out of the four that does not reach level AAA of conformance. Thus, there are more guidelines it can not check for. For instance, looking at Table 8, 1.4.6, 2.1.3, 2.4.9, 2.4.10 are level AAA success criteria, and they were never detected by WAVE. Likewise, Achecker's low guideline coverage in dynamic websites did not come as a surprise as out of all tools, it found the least number of errors in dynamic websites.

Overall, all tools had below average guideline coverage. There were many guidelines and success criteria that were never covered by any of the tools. This could mean that these websites did not fail those guidelines. It could also mean that the tools are not able to detect those guidelines and success criteria violations. However, the latter is probably the case, as it has been said that most existing tools are only able to detect about 30% of the WCAG guidelines [21,22], and that 70% of guidelines have to be checked manually. The number of success criteria detected in this experiment was 20, which is about 26% of 78, and about 33% of 61. This supports the claims stated in [21,22]. To get a more comprehensive evaluation, there is a need for human evaluation.

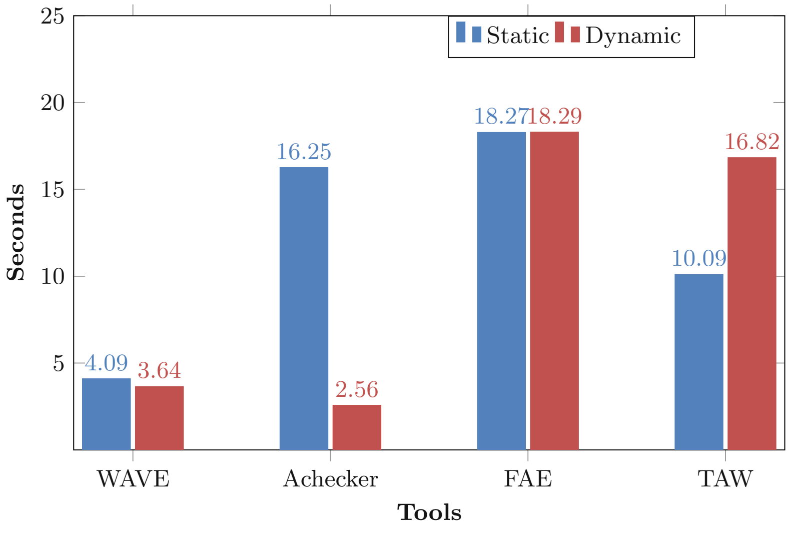


Fig. 3. Average time taken by each tool to evaluate the chosen page of each website

RQ3: How fast are the tools when evaluating websites for accessibility?
Before beginning each evaluation, we started the timer, and at the end, we stopped it and recorded the time. The time recorded is how fast the tool evaluated a single page of a website. In this section, we present the average time taken by each tool in seconds.

Result. We grouped the evaluation time into two, one for static websites and another for dynamic websites. For each tool, we calculated the average time in these two groups. Figure 3 is a bar chart showing the average time taken by each tool to evaluate a single page of a static or dynamic website. The blue bar represents the average time for static websites, and the red bar represents the average time for dynamic websites.

Looking at the figure, you can see that the tool that took the least time when evaluating static websites is WAVE, with an average time of 4.09 s. In the dynamic website category, Achecker took the least time with an average time of 2.56 s. The tool that took the most time in both dynamic and static websites is FAE, with an average time of 18.27 s in static websites and 18.29 s in dynamic websites. Additionally, the average time spent by FAE in both static and dynamic websites exceeded the time spent by all other tools in both categories.

Furthermore, from the figure, you can see that, except in the case of FAE and TAW, all other tools spent more time evaluating static websites than they did when evaluating dynamic websites. FAE took almost the same time when evaluating dynamic and static websites. It had an average time of 18.27 s when evaluating static websites, and when evaluating dynamic websites it had an average of 18.29 s. TAW took more time evaluating dynamic websites than it did evaluating static websites. When evaluating dynamic websites, the tool had an average time of 14.32 s, while it had an average time of 10.59 s when evaluating static websites.

Discussion. We observed that there is a relationship between the time it takes for the tool to evaluate a website, with the number and types of errors a tool detected, and its guideline coverage. For instance, WAVE had the least average time in the static website category. In turn, it was reported earlier that on average, WAVE detected the least number of errors and had the lowest guideline coverage in static websites. Additionally, WAVE detected the least types of errors. Similarly, the tool Achecker had the least average time of 2.67 s in dynamic websites while also having the least average number of errors and the lowest guideline coverage in dynamic websites. This further reinforces the notion that the less time it takes for the tool to evaluate websites, the less error and success criteria it detects.

On the other hand, FAE took the longest time when evaluating both static and dynamic websites. This could be due to several reasons, one of which is the reason stated prior, that the more errors and guidelines a tool can detect, the longer it takes for it to evaluate a website. This reason still holds in this case as FAE had the highest coverage overall, and it detected the second most types of errors, after TAW. Another possible reason is that in addition to WCAG 2.1

success criteria, FAE also uses HTML4, HTML5, and ARIA accessibility techniques. Hence, it had more criteria to check for than the other tools. Additionally, after FAE finishes evaluating a website, it takes extra time to save the result before presenting it. The other tools displayed the result immediately after evaluation.

Table 10. Total number of times a tool reported the same error in a website with another tool

	WAVE	Achecker	FAE	TAW
WAVE	0	18	24	23
Achecker	18	0	18	21
FAE	24	18	0	23
TAW	23	21	23	0

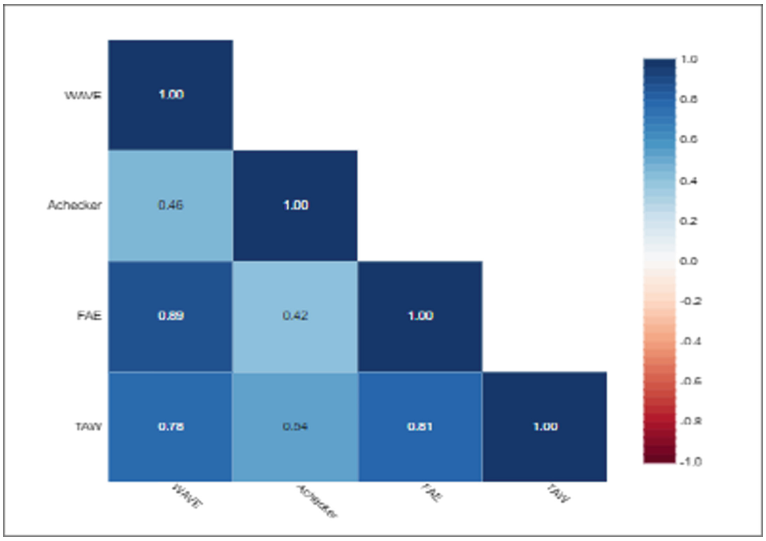


Fig. 4. Correlation matrix comparing the total number of errors found in each website per tool

RQ4: Which tools produce the most similar results?
In this subsection, we compare the tools to see which ones act the most similar. The tools can be similar to each other in the following ways: the closeness in value of the total number of errors they reported for a particular website, and if they found the same errors in a website.

Result. Table 10 shows how similar tools are in terms of the total number of times they detected the same error in a website for all 10 websites. This values in the table were calculated as follows: if two tools, say FAE and TAW, both detected the error *'image missing alternative text'* in the website *'Snap'*, a point is given to their pair. If two tools never detect the same error in any website, their pair will have 0 points. Figure 4 is a correlation matrix comparing the total number of errors reported by each tool for each website. The correlation matrix was created using a combination of Table 6 and Table 7.

From Fig. 4 and Table 10, it can be seen that the tool most similar to Wave is FAE, and vice versa. FAE and WAVE had a correlation coefficient of 0.89, which was the highest overall. In addition, the total number of times they reported the same error on a website was the highest. They reported the same error 24 times across all 10 websites. WAVE and FAE both had the least similarity with Achecker. WAVE and Achecker had a correlation coefficient of 0.46, while FAE and Achecker had a correlation coefficient of 0.42. WAVE and FAE both detected the same errors in a website with Achecker 18 times.

In contrast, the relationship between Achecker and TAW is not as symmetrical as the one WAVE has with FAE. Achecker had the highest correlation coefficient with TAW, which was 0.54. It also had the most similarity with TAW in terms of the total number of times it detected the same error in a website. They were able to detect the same errors in a website 21 times. On the other hand, TAW was most similar to FAE. They had a correlation coefficient of 0.81, and they detected the same errors in a website 23 times.

Discussion. WAVE is most similar to FAE, and vice versa. They both check against the WCAG 2.1 guideline, and this could be the reason they are most similar. Likewise, Achecker and TAW check against the same guideline, hence why Achecker had more similarity with TAW. However, the reverse is not the case between TAW and Achecker, TAW is more similar to FAE. This could be due to the fact that TAW and FAE were able to detect more types of error compared to Achecker and WAVE, hence there is a higher probability for them to detect the same error in a website.

We observed that all tools had the least similarity with Achecker. This could be attributed to the fact that Achecker had one of the lowest coverage in terms of types of errors and success criteria, compared to TAW and FAE, hence why TAW had more similarities to FAE, and not Achecker when they both checked against the WCAG 2.0 guideline. Another reason could be that Achecker produced results that were not congruent with the ones produced by the other tools.

Although there are some tools that are similar and comparable to one another, no two tools are identical. It is interesting that even tools that evaluate websites using the same guidelines and conformance level, do not produce identical results. The tools often show disparities in the types and number of errors detected in a website, and the number of elements that cause these errors. More investigation is needed to determine why this is so.

RQ5: What is the relative performance of the tools when evaluating Rich Internet Applications?
One of the reasons for selecting websites in two categories, one for static websites and the other for dynamic websites, was to observe the performance of the tools

when evaluating static websites versus when evaluating dynamic websites. In this section, we compare how the tools behaved when evaluating static websites relative to how they behaved when evaluating dynamic websites.

Result. Earlier, we presented Tables 6 and 7, which shows the total number of errors reported in static websites and dynamic websites, respectively. We also presented Fig. 3, which shows the average time taken by the tools in static and dynamic websites, and Table 9, which shows the total number of success criteria failures detected in both static and dynamic websites.

Comparing Tables 6 and 7, we can see that when evaluating static websites, most of the tools reported a higher number of errors and a higher average for the number of errors compared to when evaluating dynamic websites. For instance, the maximum number of errors detected in static websites is 211, while the maximum number of errors detected in dynamic websites is 68. Also, the maximum average of the number of errors detected in static websites was 101.8, while the maximum average in dynamic websites was 25.2. Additionally, in dynamic websites, over 50% of the time the total number of errors reported was less than 5.

A common error that was found in dynamic websites but was never found in static websites, was the error *'The language of the document is not identified or is invalid'*. This error was reported in 80% of dynamic websites and was not reported in any static website. Additionally, text errors such as the *'Fonts used'*, *'i (italic) element used'*, and *'b (bold) element used'* was found in 4 out of 5 static websites, while it was only detected in 1 out of 5 dynamic websites.

Looking at Fig. 3, you can also see that when evaluating dynamic websites versus when evaluating static websites, there is often a difference in time taken. On average, WAVE and Achecker took more time evaluating static websites than they did when evaluating dynamic websites, while FAE and TAW took more time to evaluate dynamic websites. Although, in the case of FAE, the time difference between the two categories was minuscule. The biggest disparity in time can be seen in the tool Achecker, when evaluating static websites, it had an average time of 16.25 s, while in dynamic websites, it had an average time of 2.56 s. It had over an 80% decrease in time when evaluating dynamic websites.

TAW showed the least variation in terms of the number of errors and the number of success criteria failures reported when evaluating static versus when evaluating dynamic websites. Out of all tools, the number of errors TAW reported in dynamic websites was closest to the number of errors reported in dynamic websites. Additionally, TAW detected the same number of success criteria in both static and dynamic websites. FAE showed the least variation with regard to the time taken when evaluating static websites compared to when evaluating dynamic websites. The difference in time was 20 ms, which was the least difference across all tools.

Achecker behaved the most differently when evaluating dynamic websites. It had the lowest average total number of errors in the dynamic website category, while it had the highest average total number of errors in the static website category. In addition, it was the only tool that detected no errors in 2 out of 5 dynamic websites. Additionally, there was a substantial difference in the average time it took to evaluate dynamic websites versus the time it took to evaluate static websites. The average time it took to evaluate dynamic websites was over 80% less than the time it took to evaluate static websites. There was also almost a 50% decrease in the number of success criteria failures detected in dynamic websites compared to the success criteria it detects in static websites.

Discussion. From the results above, we can see that most tools acted differently when evaluating dynamic websites compared to when evaluating static websites. The biggest difference in behavior can be seen in the total number of errors the tools reported in static websites compared to the ones reported in dynamic websites. This disparity could be due to the dynamic websites used in this experiment being more accessible than the static websites used. Alternatively, it could be that the tools are not able to evaluate dynamic web content [29]. In almost all of the dynamic websites, the error *'The language of the document is not identified or is invalid'* was reported. The fact that the language used in dynamic websites does not seem to be supported could be part of the reason the tools were not able to evaluate their contents.

Achecker showed the most disparity in the results it produced for static websites in comparison to the results it produced for dynamic sites. It had a relatively good performance in terms of error detection and guideline coverage in static websites, while it performed poorly in those aspects in dynamic websites. Furthermore, the time it took to analyze static websites was significantly greater than the time it used to analyze dynamic websites. Going by our theory that states that the longer it takes for a tool to evaluate websites, the more types of errors it can detect, we can say that Achecker was only able to detect a few errors in dynamic websites. Achecker's struggle when evaluating dynamic websites compared to when evaluating static websites could be due to the fact that it was built to evaluate static websites and not dynamic websites.

FAE is the only tool that uses the Accessible Rich Internet Applications (ARIA) guideline when evaluating websites. As we mentioned earlier, ARIA is used to provide accessibility to RIAs [31]. This could be the reason it had less discrepancy when evaluating static websites versus dynamic websites, compared to some of the other tools. TAW performed equally well, given that it does not use ARIA when evaluating websites. However, it does use HTML, CSS, and Javascript technologies, which might have helped its performance.

5.1 Limitations

While carrying out this research study, we had several limitations. Firstly, we are not experts on web accessibility, thus, we are not able to ascertain the accuracy and precision of the results presented by each tool. Although, for each error reported in a website, we checked to make sure it was valid. Another limitation we faced was the heterogeneity of the tools. They did not all check against the same guideline. In addition, the format in which they reported their results were different. This made it very difficult to compare them. Lastly, it was hard to categorize websites as static or dynamic. A lot of websites have some attributes of both. However, we tried our best to select the websites that best fit the respective categories.

6 Conclusion and Future Work

Providing accessibility to websites today has become a necessity, but sadly most websites are still not accessible. A particular type of website that poses the most challenges in regard to accessibility, are Rich Internet Applications (RIAs). Most tools are not able to evaluate RIAs due to the technologies they comprise of [12,29].

This paper presents a comparative analysis of four accessibility evaluation tools in terms of error detection, guideline coverage, speed, similarity to one another, and their relative performance when evaluating RIAs. The results of the experiment exposed the strengths and weaknesses of the tools. TAW was the tool that was able to detect the most errors. On average, WAVE was the fastest when evaluating web pages. FAE had the most guideline coverage. Additionally, the study revealed that tools are only able to cover a small fraction of the WCAG 2.0 or WCAG 2.1 guidelines and that some tools are similar to each other. Most importantly, the results showed that the tools behave differently when evaluating Rich Internet Applications versus when evaluating traditional static websites, especially the tool Achecker.

We expect our findings will provide researchers and developers with an overview of which tools are available and suitable for automatic accessibility evaluation of static and dynamic websites, their capabilities, and limitations. Through exploring the efficacy and limitations of the existing tools, developers can get a better understanding of accessibility and can thus, design websites that meet accessibility standards and tools for evaluating them. This research also highlights the remaining opportunities and research directions for better design of tools that evaluate dynamic content.

This study has some limitations which we have discussed. As a future work, we intend to work with an accessibility expert to determine the accuracy of the results produced in this experiment. Furthermore, we also plan to delve deeper into some of the issues associated with accessibility evaluation tools, to determine the root causes and come up with a viable solution.

References

1. Al-Ahmad, A., Ahmaro, I.Y., Mustafa, M.: Comparison between web accessibility evaluation tools, January 2013. https://www.researchgate.net/publication/279181298
2. Alsaeedi, A.: Comparing web accessibility evaluation tools and evaluating the accessibility of webpages: proposed frameworks. Information **11**(1), 40 (2020)
3. Baazeem, I.S., Al-Khalifa, H.S.: Advancements in web accessibility evaluation methods: how far are we? In: Proceedings of the 17th International Conference on Information Integration and Web-based Applications & Services. iiWAS 2015, 11–13 December 2015. ACM Press, New York. https://doi.org/10.1145/2837185.2843850

4. Brajnik, G., Lomuscio, R.: Samba: a semi-automatic method for measuring barriers of accessibility. In: Proceedings of the 9th International ACM SIGACCESS Conference on Computers and Accessibility, Assets 2007, October 15–17 2007, pp. 43–50. ACM Press, New York. https://doi.org/10.1145/1296843.1296853
5. Campoverde-Molina, M., Lujan-Mora, S., Garcia, L.V.: Empirical studies on web accessibility of educational websites: a systematic literature review. IEEE Access **8**, 91676–91700 (2020)
6. Fortes, R.P., Antonelli, H.L., de Lima Salgado, A.: Accessibility and usability evaluation of rich internet applications. In: Proceedings of the 22nd Brazilian Symposium on Multimedia and the Web, Webmedia 2016, 08–11 November 2016, pp. 7–8. ACM Press, New York. https://doi.org/10.1145/2976796.2988221
7. Human Network Contributor: Websites fail to comply with accessibility for people with disabilities. November 2020. https://isemag.com/2020/11/telecom-98-percent-of-websites-fail-to-comply-with-accessibility-requirements-for-people-with-disabilities/
8. Investis Digital: Accessibility testing - manual or automated? January 2020. https://www.investisdigital.com/blog/web-design-and-development/accessibility-testing-manual-or-automated. Accessed 23 Oct 2021
9. Ismail, A., Kuppusamy, K.: Web accessibility investigation and identification of major issues of higher education websites with statistical measures: a case study of college websites. J. King Saud Univ. Comput. Inf. Sci. (2019)
10. Kumar, A.: 5 free must have web accessibility testing tools - leader in offshore accessibility testing — section 508 compliance — wcag conformance — barrierbreak, October 2021. https://www.barrierbreak.com/5-free-must-have-web-accessibility-testing-tools/
11. Law Office of Lainey Feingold: Digital accessibility laws around the globe, May 2013. https://www.lflegal.com/2013/05/gaad-legal/. Accessed 23 Oct 2021
12. Linaje, M., Lozano-Tello, A., Perez-Toledano, M.A., Preciado, J.C., Rodriguez-Echeverria, R., Sanchez-Figueroa, F.: Providing ria user interfaces with accessibility properties. J. Symb. Comput. **46**, 207–217 (2011). https://doi.org/10.1016/j.jsc.2010.08.008
13. Lopes, R., Gomes, D., Carriço, L.: Web not for all: A large scale study of web accessibility. In: Proceedings of the 2010 International Cross Disciplinary Conference on Web Accessibility (W4A). W4A 2010. Association for Computing Machinery, New York (2010). https://doi.org/10.1145/1805986.1806001. https://doi.org/10.1145/1805986.1806001
14. Mankoff, J., Fait, H., Tran, T.: Is your web page accessible?: a comparative study of methods for assessing web page accessibility for the blind. In: In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI 2005, pp. 41–50. ACM Press, New York (2005)
15. Misfud, J.: 8 free web-based website accessibility evaluation tools - usability geek. https://usabilitygeek.com/10-free-web-based-web-site-accessibility-evaluation-tools/
16. Mucha, J.M.: Combination of automatic and manual testing for web accessibility. Master's thesis, Grimstad, Norway (2018). Accessed 05 Aug 2019
17. Neova Tech Solutions: 7 web accessibility testing tools you should know, April 2020. https://www.neovasolutions.com/2020/04/08/7-web-accessibility-testing-tools-you-should-know/. Accessed 03 Mar 2022
18. Smith, S.: Website accessibility standards should be higher to help disabled people - vox, May 2019. https://www.vox.com/the-goods/2019/2/5/18210912/websites-ada-compliance-lawsuits. Accessed 04 Mar 2022

19. Software Testing Help: Top 20 accessibility testing tools for web applications, Feb 2022. https://www.softwaretestinghelp.com/accessibility-testing-tools/. Accessed 03 Mar 2022
20. Timbi-Sisalima, C., Amor, C.I.M., Tortosa, S.O., Hilera, J.R., Aguado-Delgado, J.: Comparative analysis of online web accessibility evaluation tools. In: Proceedings of the 25th International Conference on Information Systems Development, ISD 2016, pp. 562–573. University of Economics, ACM Press, New York (2016)
21. Usablenet: Quick guide to manual accessibility testing and why it's important, June 2018. https://blog.usablenet.com/quick-guide-to-manual-accessibility-testing-and-why-its-important. Accessed 02 Mar 2022
22. Usablenet: Automated wcag testing is not enough for web accessibility ada compliance [blog], July 2020. https://blog.usablenet.com/automated-wcag-testing-is-not-enough-for-web-accessibility-ada-compliance. Accessed 02 Mar 2022
23. Vigo, M., Brajnik, G.: Automatic web accessibility metrics: where we are and where we can go. Interact. Comput. **23**, 137–155 (2011). https://doi.org/10.1016/j.intcom.2011.01.001
24. Vigo, M., Brown, J., Conway, V.: Benchmarking web accessibility evaluation tools: measuring the harm of sole reliance on automated tests. In: Proceedings of the 10th International Cross-Disciplinary Conference on Web Accessibility, w4A 2013, pp. 1–10. ACM Press, New York (2013). https://doi.org/10.1145/2461121.2461124
25. W3C: Web accessibility evaluation tools list. https://www.w3.org/WAI/ER/tools/
26. W3C: Web content accessibility guidelines (wcag) 2.0, December 2008. https://www.w3.org/TR/WCAG20/. Accessed 04 Mar 2022
27. W3C: Web content accessibility guidelines (wcag) 2.1, June 2018. https://www.w3.org/TR/WCAG21/. Accessed 04 2022
28. W3C Web Accessibility Initiative (WAI): Wcag 2 overview, July 2005. https://www.w3.org/WAI/standards-guidelines/wcag/. Accessed 04 Mar 2022
29. Watanabe, W.M., Fortes, R.P.M., Dias, A.L.: Using acceptance tests to validate accessibility requirements in ria. In: Proceedings of the International Cross-Disciplinary Conference on Web Accessibility, W4A 2012, 16–17 April 2012. ACM Press, New York. https://doi.org/10.1145/2207016.2207022
30. World Wide Web Consortium: Wai-aria, January 2016. https://www.w3.org/WAI/standards-guidelines/aria/. Accessed 24 Aug 2019
31. World Wide Web Consortium: Accessible rich internet applications (wai-aria) 1.1. https://www.w3.org/TR/wai-aria/. Accessed 05 Sept 2019