Catch Me If You Can: Automatically Finding Accessibility Problems in Modal Dialogs

Anonymous Author(s)

ABSTRACT

The web presents challenges for users with disabilities. Visually-impaired users navigate web content using a screen reader which makes the page audible. We conducted a user study to learn about a wide variety of web accessibility problems for people who use screen readers. Though design flaws contribute to accessibility problems, HTML does not describe the semantics necessary to drive screen readers for common user interface widgets like modal dialogs. The ARIA web markup standard can add missing semantics, but many authors neglect to add the proper tags. In this paper, we present an automated technique that can detect modal dialogs and classify whether they are accessible. We evaluated our approach on 50 popular web sites and found that 18% of them had accessibility problems, results which were validated via user study with visually-impaired users. Our approach enables authors to improve the accessibility of their web sites.

CCS CONCEPTS

•Human-centered computing → Accessibility systems and tools;

KEYWORDS

accessibility, web, ARIA, DOM analysis

ACM Reference format:
Anonymous Author(s). 2016. Catch Me If You Can: Automatically Finding Accessibility Problems in Modal Dialogs. In *Proceedings of ACM Conference, Washington, DC, USA, July 2017 (Conference'17),* 10 pages.
DOI: 10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

The web can be a frustrating place for almost all users, but it is particularly challenging for users with disabilities. For example, visually-impaired users often make use of a *screen reader*, which speaks the web page out loud to enable them to understand and navigate the content. The most popular screen readers are Jaws [20], NVDA [1], and VoiceOver [4]. The spoken text is a combination of canned fragments (e.g., speaking the word "link" for a hyperlink) and text (e.g., the label written by the web developer for a button element), all discovered by the screen reader by navigating the *accessibility tree*, a structure derived from the DOM. Screen readers can present web elements in a linear order (called *tab order*), or by enumerating all the web elements of a particular type, for example, all of the headers, links, or buttons on a page.

In theory, a web browser equipped with a screen reader should enable visually-impaired users full access to the web, but in practice, the complexity of the modern web can often interfere with full compatibility. These problems are frequently caused by the screen reader's inability to understand the semantics of all the user interface widgets created by developers.

We conducted a study to learn more about web sites that are difficult for visually-impaired people to fully access using screen reader technology. Five visually-impaired participants demonstrated to us 24 accessibility problems in 13 different categories on 20 popular web sites. Though design flaws contribute to many of the problems, some occur because HTML does not have the capacity to specify the semantics of many commonly constructed user interface widgets. These include modal dialogs, sliders, date pickers, and even menus.[1] The *ARIA* [25] web standard defines optional, semantically-oriented markup tags that developers can add to web page elements to specify their behavior and relationship to other elements. While developers *can* add these tags to their web pages to increase their accessibility, many do not, resulting in a frustrating mosaic of web sites that only partially work with screen readers.

For example, Figure 1 shows an instance of an inaccessible modal dialog which we found on the *homedepot.com* website.[2] A modal dialog is a window that users are supposed to interact with it before they can go back to the parent application. To open the dialog, a user clicks the *Create Account* button on the *Sign in* page. However, after opening the dialog, the focus remains on the main page of the document. This confuses screen readers and prevents the announcement of the new

Conference'17, Washington, DC, USA

DOI: 10.1145/nnnnnnn.nnnnnnn

[1] HTML 5.0 supports a menu element, but as of September 2018, only Firefox supports it, and only partially.
[2] Web site accessed: September 19, 2018.

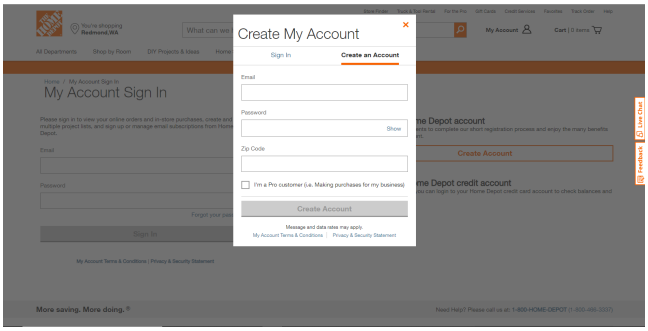


Figure 1: An inaccessible modal dialog found on *homedepot.com*.

dialog to a visually-impaired user. Worse, a user can sign in only after moving the focus with the mouse from the main page to the dialog. No keyboard support is provided, thus making the dialog inaccessible to users who rely on the keyboard for input. When we looked at the source code of the web page, we noticed that the dialog is coded as an HTML ** ** element that lacks appropriate ARIA tags, preventing accessibility tools from helping users distinguish the modal element from the rest of the web page.

To encourage improvements in accessibility, *accessibility checker* tools, such as Axe [5] and Lighthouse [10], enable web authors and developers to automatically discover accessibility issues by looking for violations of a set of heuristic rules. For example, every image on a web page should have a caption and all semantically meaningful elements should have an appropriate ARIA tag. However, all of the checkers we have seen have limited functionality. First, they make use of a narrow set of rules (to avoid false positives), but miss important concepts, such as checking for the proper ARIA tags for dialog boxes. Second, developers only run these tools once at page load time, causing them to miss web elements that are dynamically created or modified as the user interacts with the page.

In our research, we address this question: how can we help developers find and fix problems with dynamic web content, and can we do this in an automated way? In this paper, we report on our project to automatically detect problems with web-based modal dialogs. We developed a dynamic DOM analysis approach called Dialog Accessibility Checker (DAC) that can automatically analyze the DOM as it evolves. Once DAC is turned on, a user (or automated test process) drives the web site to a particular UI state where a modal dialog appears. Through a variety of heuristics, DAC detects the modal dialog classifies whether or not it is accessible.

We evaluated DAC in two ways. First, we conducted a case study in which we ran DAC on the fifty most popular web sites in five categories (shopping, news, work, government, and travel) based on the Alexa ranking of top sites [3] to detect modal dialogs and report whether they are accessible to visually-impaired users who use screen readers. (Appendix A contains the full list.) DAC found modal dialogs on 28% (14) of the web sites. It reported 13 inaccessible modal dialogs in nine of the web sites and six accessible modal dialogs in three of the web sites. Manual inspection of the web sites found that DAC missed six inaccessible modal dialogs in four of them. Second, to validate whether DAC's classifications were correct, we conducted a user study in which four visually-impaired participants were asked to try out a sample of modal dialogs from six of the web sites. In all evaluated cases, the participants confirmed that DAC's classifications were correct. Afterwards, we notified the web site authors of the problems found by DAC. We hope they are fixed soon.

Our work makes the following contributions:

- We performed a qualitative user study to learn about the most pressing accessibility issues in web site navigation for visually-impaired people using screen readers.
- We have implemented an approach, *Dialog Accessibilty Checker* (DAC), that monitors a web page's DOM as it evolves during the navigation of a web site, and heuristically identifies modal dialogs. DAC detects them when they appear statically within the web page, and also when they are created dynamically.
- DAC automatically analyzes modal dialogs for accessibility errors that prevent them from being used by visually-impaired people using screen readers. The prior state-of-the-art in accessibility checkers cannot find these problems.
- We validated DAC, both through a case study of its performance and, most importantly, by a study that ensured that visually-impaired users agree qualitatively with DAC's accessibility classification.

2 RELATED WORK

There has been extensive prior work on studying accessibility problems on the Web. WebinSitu [7] is a comparative study of browsing experiences between blind and sighted users. One of the observations is that blind people avoid visiting pages that suffer from severe accessibility problems, such as those related to dynamic content. Furthermore, Power *et al.*'s study [18] suggests that only half of the accessibility problems are covered by existing accessibility guidelines. Aizpura *et al.* [2] supports this observation, further highlighting the lack of content and functionality expectations in accessibility guidelines.

Bigham *et al.* reports that many users who need accessibility tools do not know which information is available, and which is available, but inaccessible [6]. Investigating further into how accessibility problems affect visually-impaired users, Vigo *et al.*'s study found that one of the navigation

tactics screen reader users must learn is how to escape from inaccessible content [23]. Voykinska *et al.* found that complicated page structures and scrolling feeds that are incompatible with screen readers cause further accessibility problems [24]. Things are slowly getting better, however, due to changes in coding practices and website technologies [12, 19].

Mankoff *et al.* reported on several methods for detecting accessibility issues on web sites [15]. Since then, several popular web accessibility evaluation tools, including WAVE [26], AXE [5], and Lighthouse [10], have helped developers determine if web content meets accessibility guidelines. The tools' approaches make use of static DOM analyses to check whether a web page satisfies accessibility best practice heuristics. More recent research approaches have helped developers discover missing "alt" attributes in images [11, 27], verify whether their web page has an accessible layout [17], predict headings on web pages [8], and make the web page display larger without causing negative side effects [6].

Though accessibility checkers mainly rely on statically analyzing web pages, our work developing a dynamic DOM analysis has been inspired by several other recent web analysis tools. Liang *et al.*'s SeeSS system automatically tracks CSS changes across a site and enables developers to easily visualize all of them together [14]. Burg *et al.*'s Scry tool discovers DOM changes caused by dynamic interactions and localizes them to the Javascript code responsible [9]. Silva *et al.* propose a reverse engineering approach based on static and dynamic code analysis for extracting information about the structure of the user interface as well as its behavior [21]. Another approach based on program analysis is Genie [22], a system that automatically infers an abstract model of the underlying commands in a web application, enabling interaction with the web page through alternative interfaces (e.g., a keyboard). Finally, Ko and Zhang's Feedlack tool helps developers discover when their web site fails to provide visual feedback after a user interacts with the web page (e.g., receiving no feedback about success or failure after submitting a form) [13].

3 STUDY 1

We conducted a study to identify the most common accessibility problems visually-impaired users using screen readers have when interacting with web content. It is important to note that while those with full vision *can* make use of screen readers, their lack of expertise makes it difficult to understand the true difficulties that expert low vision and non-sighted users encounter when browsing the web sites with screen readers.

Method

The study consisted of an single interview session lasting between 45 minutes and one hour. Our study design was approved by *Anonymized organization*'s IRB.

Participants. We recruited five visually-impaired participants who had experience using screen readers to navigate web pages. We found them by advertising on a mailing list for visually-impaired employees at *Anonymized organization*. Of the five participants, one has low vision, while the other four describe themselves as blind. All of the participants were male. Four participants were full-time employees at *Anonymized organization*, and one was a summer intern. Each participant was compensated with an Amazon gift certificate for USD $20.

Participant	Browser	Screen Reader	Experience	Sites
P1	Chrome	JAWS	22 yrs.	2
P2	IE	JAWS	8 yrs.	6
P3	Chrome/IE	JAWS	5 yrs.	10
P4	Chrome	JAWS	20 yrs.	4
P5	Chrome	NVDA	26 yrs.	5

Table 1: Participants in the study and the technology they use to browse the web. Experience indicates the number of years the participant has used screen readers to navigate the web. The final column is the number of web sites participants showed us during their study session.

Table 1 shows the screen reader and browser which our participants typically use, as well as how long they have used a screen reader. Four participants use JAWS [20], a popular commercial screen reader, while one uses NVDA [1], a free screen reader used by a community of over 70,000 users. Three participants use the Chrome browser, one uses Internet Explorer, and one uses both.

Task. We asked participants to perform two tasks. First, we asked them to demonstrate to us a set of web sites which they considered to be accessible. Second, we asked them to show us a set of web sites which they found inaccessible. We asked them to explain each accessibility issue they found, and to help us understand how to reproduce them. Table 1 also shows how many total web sites each participant showed us.

Findings

Overall, participants showed us 20 unique web sites (both accessible and not) with 32 instances of accessibility issues. Table 2 illustrates the distribution of accessibility issues in the web sites that were visited. Notice that almost every web site suffers from accessibility problems, even sites that were

Website	# Issues
Anonymized organization home page	4
Anonymized organization code repository	4
bitbucket.com	3
udacity.com	3
priceline.com	2
twitter.com	2
Anonymized organization banking site	2
bbc.com	1
bing.com	1
cnn.com	1
doc.microsoft.com	1
engadget.com	1
github.com	1
google.com	1
msdn.microsoft.com	1
quora.com	1
stackoverflow.com	1
walmart.com	1
wikipedia.org	1
youtube.com	0
Total	24

Table 2: Accessibility issues reported per web site visited ordered by the number of issues reported.

Issue Category	Web Sites
Too much content	5
Active element without focus	4
Complex widgets without semantics	4
Widgets are not keyboard accessible*	4
Missing or bad labels+	3
Application vs browse mode	2
Bad layouts*	2
Dynamic change is not announced	2
Missing or few headings	2
Distracting alerts	1
Distracting videos	1
No search area	1
Too many blank spaces	1

Table 3: Categories of issues reported in our study. Existing tools can find missing or bad labels (+), but can only partially find widgets that are not keyboard-accessible or pages with bad layouts (*). No other issues can be found by existing tools.

Figure 2: Overview of the Dialog Accessibility Checker

deemed mostly accessible! On a brighter note, no participants reported any problems with *youtube.com*.

We grouped the issues into 13 non-mutually exclusive categories (shown in Table 3) based on the type of problems reported by the participants. The *only* one fully addressed by existing accessibility checker tools is that ability to find missing or bad labels.

Too much content on a web page is the most frequent category of accessibility problem. Examples include content with too many headings, or content with too many visual elements to comprehensively navigate. Other very common problems are *widgets without semantics*, *active element without focus* and *widgets that are not keyboard accessible*. The first problem is common among user interface widgets, such as menus, modal dialogs, menu buttons, and sliders, which are typically coded with non-semantic HTML elements, such as ** ** and ****. As a result, screen reader users have difficulties understanding and using them. Participants also reported accessibility problems related to an *active element without focus* on 4 of the web pages. A typical example included opening a modal dialog without it receiving focus. This prevents the screen reader from noticing the dialog's content, rendering it inaccessible. Finally, it was also reported that some elements on the web pages were programmed to only respond to mouse events, preventing use by keyboard. One example of this kind of element was a date picker which could only be opened by mouse.

We can alternatively group the reported accessibility issues into two categories: *design choices* and *coding errors*. The first category refers to issues that are related to the design of a web page, such as layout, color contrast, number of UI elements, and the type of elements presented on a web page. The second group consists of problems related to inadequate or improper use of HTML elements or ARIA markup. For example, ** ** elements programmed to behave like buttons are inaccessible when their semantics is unknown to screen readers.

Our findings strongly motivate the need for approaches and tools that can help developers make the content of their web pages accessible to all users.

4 OUR APPROACH

In this section we present our technique, Dialog Accessibility Checker (DAC), to detect inaccessible modal dialogs in websites. Figure 2 illustrates the main components in DAC used to detect inaccessible modal dialogs. First, the *Tracking Changes* component tracks dynamic changes on web page, specifically, whether a new element is added to a web page

or the visibility of existing element is changed. Second, *DOM Analysis* performs a static check on new element to determine is an element a modal dialog. Static checks are based on a set of heuristics we later define in this section. Finally, if a modal dialog is created, *Check Accessibility* checks the accessibility properties of the modal dialog.

DAC is implemented as a Chrome extension. It can be used either during manual or automated testing of Web applications. Our implementation is based on an online analysis of dynamic DOM changes and builds on the MutationObserver API (supported by all major browsers). DAC outputs its results into a log, reporting any detected HTML elements representing the inaccessible modal dialogs and the results of its accessibility checks.

Tracking Dynamic Changes

To detect dynamic changes on website, we use the *MutationObserver* [16], provided by the Web API. The *MutationObserver* tracks changes in the DOM, such as adding new elements, removing existing elements or changing attribute values. The approach finds new modal dialogs by monitoring the addition of new DOM elements that are visible in the viewport and changes in the visibility of existing elements. The viewport represents the screen area visible to the user. It is important to check for the visibility of new elements because modal dialogs require focus only when they become visible. To determine when the visibility of an existing element changes, DAC tracks changes in the *style* attribute. More specifically, we say that an element becomes visible if one of the following holds:

- *Display* property of the style attribute changes value from "none" to a different value.
- *Visibility* property of the style attribute changes value from "hidden" to "visible".
- *Opacity* property of the style attribute changes value from less than or equal to zero to become strictly positive.

After looking through many web sites with modal dialogs, we believed that most were the result of a single DOM change. However, during the course of this work we found that some dialogs are coded by triggering several DOM changes. For example, one change might change the background color of the body element, while the other changes the visibility of the dialog box element. Precisely discovering when multiple DOM changes are working together to create a modal dialog is a much more difficult problem to solve than detecting modal dialogs created by single DOM changes. We leave this more complex case for future work.

DOM Analysis

The key characteristic of a modal dialog is that it gets an immediate focus by "dimming" the rest of the web page's content. We pragmatically define modal dialog, as follows.

Definition 4.1 (Modal dialog). A modal dialog d is a DOM element with the following properties:

- d is coded as a ** ** HTML element
- d contains one or more elements such as: *buttons*, *headings*, *links*, *inputs*, *tables* or *text*.
- d is an *overlay element*.
- when d is open, its background color is different than the background color of *body* element.

An element, e, is an overlay element, meaning that it covers the entire page, if e's width is at least the window's width, if e's height is at least the window's height and if e's top and left margins are equal to zero. The last property in the definition ensures that the rest of the page gets dimmed when the dialog is open.

Accessibility Checks

HTML has a limited set of UI elements that have defined semantics. To provide more specific meanings to generic tags, such as ** **, the ARIA standard proposes the use of the *role* attribute. After DAC detects a modal dialog, it checks whether a dialog has a *role* attribute. If a *role* attribute is not defined, or different than "dialog," DAC reports a warning. Furthermore, DAC checks if the current focus is on the modal dialog. If this is not the case, the screen reader user would not be able to consume the content contained in the dialog.

5 EVALUATION

To evaluate DAC's effectiveness and efficiency, we asked the following questions:

(1) *Can DAC find inaccessible modal dialogs?*
(2) *How often does DAC miss inaccessible modal dialogs?*
(3) *How often does DAC report false positives?*
(4) *Do visually-impaired users agree with DAC's classification of accessibility?*

We answer the first three questions using a case study approach, in which we use DAC to check 50 popular web sites for the accessibility of their modal dialogs (if any). We answer the fourth question in a user study with visually-impaired users, well-experienced with screen readers, who confirmed the DAC's classification results on a sample of the same 50 web sites.

Case Study

We checked 50 popular web sites for accessibility problems in modal dialogs. First, we selected five categories of sites (*shopping*, *news*, *work*, *government* and *travel*) based on our understanding that the accessibility of these kinds of web sites are very important. In each category, we chose the top ten most popular web sites, based on the Alexa ranking of top sites [3]. The full list of web sites is in Appendix A.

Method. We manually executed the DAC technique on each web site using the following procedure:

(1) We allow a ten minute time budget for exploration of each website.
(2) On each web site, we explore the main page by performing the following actions:
    (a) click on all buttons and links found on the page
    (b) perform the search operation, if available
    (c) fill and submit any available form
    (d) scroll down to the bottom of the page.
    During the interaction with the website, if DAC discovers an inaccessible dialog, it marks it and logs the source of the likely cause: either missing focus or lack of ARIA markup.
(3) Continue exploring other pages within the same web site until the time runs out, however we skip pages that require login or user registration.

We performed this procedure in the Chrome browser, and used the DevTools feature to methodically check for true positives, false positives, and false negatives. After each new modal dialog opens, we manually inspect the corresponding HTML element for two properties: focus and ARIA markups. If DAC marks a dialog as inaccessible, and it does not have the *role* attribute and focus, we confirm it as a true positive. If DAC misses such a dialog, we label it as a false negative. Finally, if DAC reports a dialog as inaccessible, yet it has both a *role* attribute and focus, we say that the dialog represents a false positive.

Results. Overall, we found that 14 of the 50 web sites in our corpus had modal dialogs. In Table 4, we list each of these fourteen web sites along with the number of modal dialogs that DAC classified as accessible and inaccessible. For each, we also quantify how many modal dialogs our technique detected and how many it missed. In total, DAC finds 13 accessibility problems in 9 popular websites, but misses 6 inaccessible modal dialogs in 4 websites. In addition, it detected 6 modal dialogs in 3 websites and labeled them as accessible.

Examples. Figure 3a illustrates an example of a reported inaccessible modal dialog on *reddit.com*. Reddit.com aggregates social news and is the most popular website in the News category in Alexa's global ranking. After clicking on the *log in* button, a modal dialog opens which DAC reports has neither a focus nor an ARIA *role* attribute. Moreover, after dialog opens, the focus remains on the main document, and the user can only use a mouse to move the focus to the dialog.

Similarly to the previous example, Figure 3b illustrates the reported inaccessible dialog on *mindtools.com* website. Mindtools.com provides skill-building resources and ranks as the top website in the Work category in Alexa's global ranking of web sites. Found on the *sign up* page, the modal dialog is activated by clicking on *Get Standard Membership* button. Once open, the modal dialog does not receive focus, nor does it have an ARIA *role* attribute.

Website	Category	Reported inaccessible		Reported accessible	
		D	M	D	M
agoda.com	travel	2	0	0	0
auspost.com.au	govt.	0	1	0	0
delta.com	travel	0	0	1	0
etsy.com	shopping	0	0	2	0
homedepot.com	shopping	2	1	0	0
hotels.com	travel	0	0	3	0
indiatimes.com	news	1	0	0	0
mindtools.com	work	2	0	0	0
news.yahoo.com	news	1	0	0	0
reddit.com	news	2	0	0	0
southwest.com	travel	1	2	0	0
tripadvisor.com	travel	0	2	0	0
wapo.com	news	1	0	0	0
who.int	govt.	1	0	0	0
Total:		13	6	6	0

Table 4: Web sites with reported inaccessible and accessible modal dialogs. D denotes detected dialogs, M denotes missed dialogs.

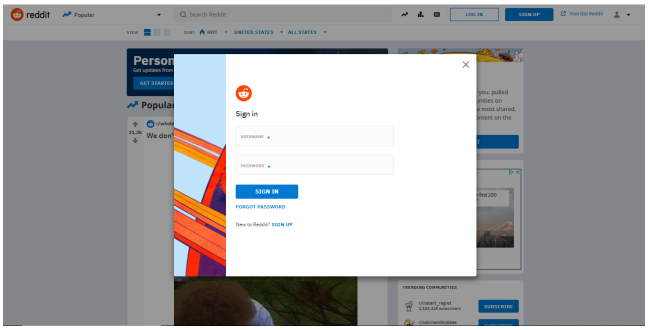
Who.int is the website of World Health Organization: it is ranked 6th under the Government category in Alexa top ranking. As shown in Figure 3c, the web site's search button opens a modal dialog without an ARIA *role* attribute. After opening, the dialog does not receive focus. In addition, we found that the search button that opens the dialog is not keyboard accessible, making it inaccessible to users of screen readers.

Study 2

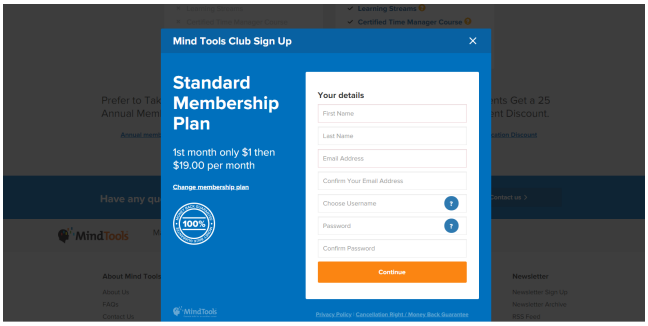
In order to validate DAC's findings, we performed a qualitative study with four visually-impaired employees at *Anonymized organization*.

Method. We worked with each participant for 30 minutes. Each was compensated with an Amazon gift certificate worth USD $20. Our study methodology was approved by *Anonymized organization*'s IRB.

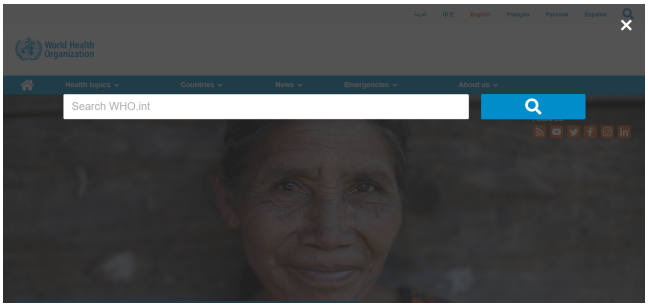
Participants. We recruited participants by advertising the study on a mailing list for visually-impaired employees at *Anonymized organization* (the same one as in Study 1). Four people responded to our advertisement and joined the study. Participants P2 and P4 also participated in Study 1. All participants were full-time employees at *Anonymized organization*



(a) Inaccessible modal dialog on reddit.com.



(b) Inaccessible modal dialog on mindtools.com.



(c) Inaccessible modal dialog on who.int.

Figure 3: Inaccessible modal dialogs found by DAC. All were found to lack focus and an ARIA *role* attribute.

and all had extensive experience using screen readers to navigate web content. One participant has low vision, while the other three describe themselves as blind. Three out of four participants were male and one was female. Table 5 shows which screen reader and browser our participants used to complete tasks in the study, along the number of years they have used screen readers. It also shows how many web sites each participant visited during the study session.

Task. We asked each participant to visit three websites and perform navigation tasks that would result in the creation of modal dialogs. Two participants were asked to complete each task. Table 6 details the navigation task for each web site.

After each task, we asked the following questions:

Participant	Browser	Screen Reader	Experience	Sites
P1	IE	JAWS	40	3
P2	IE	JAWS	8	3
P3	Edge	NVDA	15	2
P4	Chrome	JAWS	20	3

Table 5: Participants in the second user study and technology they use to browse the web. Experience indicates the number of years the participant has used screen readers to navigate the web. The final column is the number of web sites participants the participants were able to visit during their study session. P2 and P4 also participated in Study 1.

- Did you find the element required to complete the task (e.g., button or link)?
- What was the result of the click action on the element?
- Did you understand what editable fields were present in the dialog?
- If yes, were you able to fill in those fields?
- Were you able to exit the modal dialog?

If the user was not aware that a modal dialog appeared on the web page, we told them it was there before asking the final two questions.

Findings. In total, participants assessed the accessibility of six modal dialogs. All of the dialogs DAC reported to be accessible were confirmed by the participants to be accessible. All of the dialogs DAC reported to be inaccessible were also confirmed by the participants to be inaccessible. Below, we present more detail on each participant's experiences during the study.

delta.com. P3 could not complete the task because she did not find the search button; P2 easily found it. Once P2 executed the search button, the screen reader announced a new dialog. P2 understood and filled in the editable fields and successfully closed the dialog with the *Esc* key. P2 confirmed that the dialog was accessible.

etsy.com. Both participants, P1 and P4 confirmed that the dialog created on sign up link was accessible because their screen readers announced it. They said that all of the editable fields were easy to understand and fill in. They were both able to closed the dialog with the *Esc* key.

agoda.com. Both participants found the *create account* button. P3 could not tell what happened after clicking on the button, could not edit the fields, and could not close the dialog. P1 assumed that a dialog would result from clicking on the button, even though his screen reader did not announce it.

Website	Task	Confirmed Accessible	Confirmed Inaccessible
delta.com	Click on search button	x	
etsy.com	Click on sign up link	xx	
agoda.com	Click on create account button		xx
homedepot.com	Click on my account link and then on sign in link		xx
reddit.com	Click on sign up link		x
mindtools.com	Click on sign up link and then on get standard membership button		xx

Table 6: Navigation tasks for the user study along with whether participants confirmed their modal dialogs to be accessible or not. Two participants were unable to complete one of their tasks.

He was not able to close the dialog box with the *Esc* key. Both participants confirmed that the dialog was inaccessible.

homedepot.com. Both participants successfully found the *my account* link and the *sign in* link. P1 could access the set of editable fields, but found them difficult to understand because they did not have labels. The screen reader did not announce the dialog, nor was the dialog closeable with the *Esc* key. P3 could not access editable fields because the focus was not on the dialog. Both participants confirmed that the dialog was inaccessible.

reddit.com. P4 could not complete the task because she did not find sign up link. Though P2 did find the link, the screen reader did not announce the modal dialog. Furthermore, the dialog did not get focus, but was closeable via the *Esc* key. P2 confirmed that the dialog was inaccessible.

mindtools.com. Both P2 and P4 easily found the *get standard membership* button, but they could not access the dialog because it lacked focus. Their screen readers did not announce the dialog. In addition, they were unable to close the dialogs with the *Esc* key. Both participants confirmed that the dialog was inaccessible.

Due to the limited amount of time we had with each participant and the small number of participants who volunteered, we were only able to have the participants confirm the accessibility of just six of the web sites in the Case Study corpus. Because of this, we confirmed the rest of the sites through manual inspection of their source code. The other dialogs classified by DAC as inaccessible share the same properties as confimed ones: they have neither focus nor the ARIA *role* attribute (or have just one). The other dialogs classified by DAC as accessible all receive focus, have an ARIA *role* attribute equal to "dialog," and were all closeable with the *Esc* key.

Evaluation Summary

Here we summarize the answers to our questions.

(1) *Can DAC find inaccessible modal dialogs?* In total, DAC finds 13 suspected inaccessible modal dialogs across 9 out of 50 websites.
(2) *How often does DAC miss inaccessible modal dialogs* DAC misses 6 instances of suspected inaccessible dialogs from 4 websites.
(3) *How often does DAC report false positives?* DAC reported no false positives.
(4) *Do visually-impaired users agree with DAC's classification of accessibility?* Participants in our study confirmed that suspected inaccessible dialogs were impossible or difficult to navigate, and that suspected accessible dialogs were easy.

We have reported each of the confirmed inaccessible dialogs to the respective websites and hope that they fix them soon.

6 THREATS TO VALIDITY

There are two main limitations in our DAC approach. To find accessibility issues in web applications, we have to manually run through the user interface on each web page. This means that DAC may not detect all instances of inaccessible dialogs. Our approach would be more complete with an automated tool that performs exhaustive UI testing of web pages. We are not aware of such UI testing tools, and since the testing problem is the orthogonal to our approach, we leave it for future work.

Second, modal dialogs can be coded in many different ways. Our approach does not detect dialogs resulting from multiple DOM changes. However, our evaluation shows that our approach finds more instances of inaccessible modal dialogs than it misses. This proves the effectiveness of our heuristics in practice.

7 FUTURE WORK

We plan to pursue several directions in the future to address more of the issues we reported in Table 3. In particular, we would like to improve the accessibility of other complex widgets, such as menus or date pickers. The participants in our first study reported that *menus* were a source of problems

because they lacked well-defined semantics. Although the ARIA standard defines a menu as a widget that "offers a list of choices to the user" and provides *menu* and *menubar* values for the *role* attribute, we found that visually-impaired users themselves disagree on which elements on a web page should be considered to be menus and behave in any particular way. In addition, we found that screen readers do not always respect these ARIA roles.

Beyond further study of the impact of dynamically-changing web pages on accessibility, we believe that static analysis tools for web pages should be improved to help detect accessibility problems without the need to "execute" the web page.

8 CONCLUSION

In this paper, we presented an exploratory study of real-world accessibility issues. We categorized the problems reported by our participants and distinguished between two primary sources of accessibility issues: *design choices* and *coding errors*. We developed a technique, Dialog Accessibility Checker, to detect modal dialogs and classify their accessibility to help web authors and developers fix some accessibility issues in their code. Our case study shows the effectiveness of the DAC approach, and our user study confirms the correctness of its classifications on many popular websites.

Websites are not as accessible as they could or should be. The ARIA markup standard addresses some of the accessibility challenges, but visually-impaired users must still deal with many others. The trend towards dynamic pages has made it more difficult to check to see if a web site successfully meets the ARIA standard, offering plenty of opportunity for future research to further improve web accessibility. We hope our results and observations motivate future work in this very promising research area.

REFERENCES

[1] NonVisual Desktop Access. 2018. NVDA screen-reader. https://www.nvaccess.org/. (2018). Retrieved September 20, 2018.

[2] Amaia Aizpurua, Myriam Arrue, and Markel Vigo. 2013. Uncovering the Role of Expectations on Perceived Web Accessibility. In *Proceedings of the 15th International ACM SIGACCESS Conference on Computers and Accessibility (ASSETS '13)*. ACM, New York, NY, USA, Article 74, 2 pages. DOI: http://dx.doi.org/10.1145/2513383.2513411

[3] Alexa. 2018. The top 500 sites on the web. https://www.alexa.com/topsites. (May 2018). Retrieved September 10, 2018.

[4] Apple. 2018. VoiceOver. https://www.apple.com/accessibility/mac/vision/. (2018). Retrieved September 20, 2018.

[5] Axe-Core. 2018. Axe: The Accessibility Engine. https://www.deque.com/axe/. (2018). Retrieved September 18, 2018.

[6] Jeffrey P. Bigham. 2014. Making the Web Easier to See with Opportunistic Accessibility Improvement. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology (UIST '14)*. ACM, New York, NY, USA, 117–122. DOI: http://dx.doi.org/10.1145/2642918.2647357

[7] Jeffrey P. Bigham, Anna C. Cavender, Jeremy T. Brudvik, Jacob O. Wobbrock, and Richard E. Ladner. 2007. WebinSitu: A Comparative Analysis of Blind and Sighted Browsing Behavior. In *Proceedings of the 9th International ACM SIGACCESS Conference on Computers and Accessibility (Assets '07)*. ACM, New York, NY, USA, 51–58. DOI: http://dx.doi.org/10.1145/1296843.1296854

[8] Jeremy T. Brudvik, Jeffrey P. Bigham, Anna C. Cavender, and Richard E. Ladner. 2008. Hunting for Headings: Sighted Labeling vs. Automatic Classification of Headings. In *Proceedings of the 10th International ACM SIGACCESS Conference on Computers and Accessibility (Assets '08)*. ACM, New York, NY, USA, 201–208. DOI: http://dx.doi.org/10.1145/1414471.1414508

[9] Brian Burg, Andrew J. Ko, and Michael D. Ernst. 2015. Explaining Visual Changes in Web Interfaces. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology (UIST '15)*. ACM, New York, NY, USA, 259–268. DOI: http://dx.doi.org/10.1145/2807442.2807473

[10] Tools For Web Developers. 2018. Lighthouse. https://developers.google.com/web/tools/lighthouse/. (2018). Retrieved September 18, 2018.

[11] Darren Guinness, Edward Cutrell, and Meredith Ringel Morris. 2018. Caption Crawler: Enabling Reusable Alternative Text Descriptions Using Reverse Image Search. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18)*. ACM, New York, NY, USA, Article 518, 11 pages. DOI: http://dx.doi.org/10.1145/3173574.3174092

[12] Vicki L. Hanson and John T. Richards. 2013. Progress on Website Accessibility? *ACM Trans. Web* 7, 1, Article 2 (March 2013), 30 pages. DOI: http://dx.doi.org/10.1145/2435215.2435217

[13] Andrew J. Ko and Xing Zhang. 2011. Feedlack Detects Missing Feedback in Web Applications. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '11)*. ACM, New York, NY, USA, 2177–2186. DOI: http://dx.doi.org/10.1145/1978942.1979260

[14] Hsiang-Sheng Liang, Kuan-Hung Kuo, Po-Wei Lee, Yu-Chien Chan, Yu-Chin Lin, and Mike Y. Chen. 2013. SeeSS: Seeing What I Broke – Visualizing Change Impact of Cascading Style Sheets (Css). In *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology (UIST '13)*. ACM, New York, NY, USA, 353–356. DOI: http://dx.doi.org/10.1145/2501988.2502006

[15] Jennifer Mankoff, Holly Fait, and Tu Tran. 2005. 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 (CHI '05)*. ACM, New York, NY, USA, 41–50. DOI: http://dx.doi.org/10.1145/1054972.1054979

[16] Mozilla. 2018. Mutation Observer. https://developer.mozilla.org/en-US/docs/Web/API/MutationObserver. (2018). Retrieved September 18, 2018.

[17] Pavel Panchekha, Adam T. Geller, Michael D. Ernst, Zachary Tatlock, and Shoaib Kamil. 2018. Verifying That Web Pages Have Accessible Layout. In *Proceedings of the 39th ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI 2018)*. ACM, New York, NY, USA, 1–14. DOI: http://dx.doi.org/10.1145/3192366.3192407

[18] Christopher Power, André Freire, Helen Petrie, and David Swallow. 2012. Guidelines Are Only Half of the Story: Accessibility Problems Encountered by Blind Users on the Web. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '12)*. ACM, New York, NY, USA, 433–442. DOI: http://dx.doi.org/10.1145/2207676.2207736

[19] John T. Richards, Kyle Montague, and Vicki L. Hanson. 2012. Web Accessibility As a Side Effect. In *Proceedings of the 14th International ACM SIGACCESS Conference on Computers and Accessibility (ASSETS '12)*. ACM, New York, NY, USA, 79–86. DOI: http://dx.doi.org/10.1145/2384916.2384931

[20] Freedom Scientific. 2013. JAWS for Windows® Screen Reading Software. *Consulté le* 17 (2013).

[21] Carlos E. Silva and José C. Campos. 2013. Combining Static and Dynamic Analysis for the Reverse Engineering of Web Applications. In *Proceedings of the 5th ACM SIGCHI Symposium on Engineering Interactive Computing Systems (EICS '13)*. ACM, New York, NY, USA, 107–112. DOI: http://dx.doi.org/10.1145/2494603.2480324

[22] Amanda Swearngin, Andrew J. Ko, and James Fogarty. 2017. Genie: Input Retargeting on the Web Through Command Reverse Engineering. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17)*. ACM, New York, NY, USA, 4703–4714. DOI: http://dx.doi.org/10.1145/3025453.3025506

[23] Markel Vigo and Simon Harper. 2013. Challenging Information Foraging Theory: Screen Reader Users Are Not Always Driven by Information Scent. In *Proceedings of the 24th ACM Conference on Hypertext and Social Media (HT '13)*. ACM, New York, NY, USA, 60–68. DOI: http://dx.doi.org/10.1145/2481492.2481499

[24] Violeta Voykinska, Shiri Azenkot, Shaomei Wu, and Gilly Leshed. 2016. How Blind People Interact with Visual Content on Social Networking Services. In *Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social Computing (CSCW '16)*. ACM, New York, NY, USA, 1584–1595. DOI: http://dx.doi.org/10.1145/2818048.2820013

[25] WAI-ARIA. 2018. Accessible Rich Internet Applications suite of web standards. https://www.w3.org/WAI/standards-guidelines/aria/. (2018). Retrieved September 18, 2018.

[26] WebAIM. 2018. WAVE Web Accessibility Tool. http://wave.webaim.org/. (2018). Retrieved September 18, 2018.

[27] Shaomei Wu, Jeffrey Wieland, Omid Farivar, and Julie Schiller. 2017. Automatic Alt-text: Computer-generated Image Descriptions for Blind Users on a Social Network Service. In *Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing (CSCW '17)*. ACM, New York, NY, USA, 1180–1192. DOI: http://dx.doi.org/10.1145/2998181.2998364

A APPENDIX

Websites used for the evaluation

Shopping.

(1) amazon.com
(2) ebay.com
(3) walmart.com
(4) etsy.com
(5) ikea.com
(6) store.steampowered.com
(7) bestbuy.com
(8) homedepot.com
(9) target.com
(10) nike.com

News.

(1) reddit.com
(2) news.google.com
(3) cnn.com
(4) nytimes.com
(5) theguardian.com
(6) indiatimes.com
(7) foxnews.com
(8) news.yahoo.com
(9) washingtonpost.com
(10) weather.com

Work.

(1) mindtools.com
(2) iLo.com
(3) flexjobs.com
(4) dol.gov
(5) virtualvocations.com
(6) brighthorizons.com
(7) ethicspoint.com
(8) contractoruk.com
(9) diversityinc.com
(10) contractcalculator.co.uk

Government.

(1) europa.eu
(2) state.gov
(3) who.int
(4) un.org
(5) royalmail.com
(6) worldbank.org
(7) fao.org
(8) auspost.com.au
(9) gov.uk
(10) unesco.org

Travel.

(1) booking.com
(2) tripadvisor.com
(3) expedia.com
(4) hotels.com
(5) agoda.com
(6) xe.com
(7) kayak.com
(8) delta.com
(9) southwest.com
(10) aa.com