Making Websites More Accessible For Blind People With Automatic HTML Code Transformations

Ana Ferreira[1], **Márcio Ribeiro**[1]

[1]Computing Institute – Federal University of Alagoas (UFAL)
Maceió – AL – Brazil

{alsf,marcio}@ic.ufal.br

Abstract. *The Brazilian Inclusion Law (LBI) establishes mandatory accessibility rules on all websites. However, only 0.74% of the Brazilian websites meet the accessibility rules. Existing tools are capable of identifying accessibility problems (e.g., lack of images descriptions, lack of language information etc) when considering blind people that use screen readers to navigate on websites. Nevertheless, these tools do not provide transformations to remove the problems. To minimize this scenario, in this paper we introduce a preliminary catalog and a Visual Studio Code IDE plugin to make websites more accessible. The catalog presents some transformations on HTML code and the plugin implement them. We evaluate our catalog and plugin in a two-fold empirical study: with online evaluators, that are capable of scoring HTML code in terms of accessibility; and with eight blind people. The results show that the catalog and plugin can reduce the number of critical errors in terms of accessibility and that the majority of the blind participants was able to identify our improvements.*

1. Introduction

According to the W3C (World Wide Web Consortium), a website is accessible when it ensures that people with different abilities and needs can perceive, understand, navigate, and interact with the web effectively [Consortium 2013]. This means that the site offers the necessary resources and tools so that these people can navigate autonomously and comfortably.

In this context, the Brazilian Inclusion Law (LBI) establishes mandatory accessibility rules on all websites, which should impact positively on blind people that use assistive technologies, such as screen readers that help navigating in websites. However, previous studies [All 2020] show that only 0.74% of the Brazilian websites meet the accessibility rules. In particular, the websites suffer from common problems, such as the lack of language information, lack of alternative texts to describe the images, lack of a better structure to better describe tables etc. Existing tools [Systems 2021, van der Schee 2018] are capable of identifying these problems, but unfortunately they do not transform the HTML code to remove them.

To minimize these problems, in this paper we introduce a preliminary catalog of transformations in HTML code to make websites more accessible. The catalog contains four transformations and rely on pattern matching. Each transformation is composed by two sides. There is a "problematic" left-hand side and a right-hand side with the "refactored" version, which adds HTML tags and attributes to improve the code in terms of accessibility. We implement this catalog in terms of a plugin for the popular Visual Studio

Code IDE. Thus, given a code that matches the left-hand side of one of our transformations, the plugin automatically converts the code to the proposed right-hand side.

To evaluate our catalog and tool, we focus on two research questions: **RQ1:** To what extent our catalog makes websites more accessible when considering the standards from the W3C? **RQ2:** To what extent blind people perceive the improvements provided by our tool? To answer them, we rely on a two-fold empirical study. The first part answers **RQ1** by using three online accessibility evaluators. We submit three websites (i.e., *Prefeitura de São Paulo*, *Banco Inter*, and *Portal da Transparência da Prefeitura de Maceió*) to these evaluators before and after applying our plugin in their HTML code and collect the number of critical errors. The second part answers **RQ2**, where we conduct an online survey based on the Latin Square Design [George E. P. Box and Hunter 2005] with eight blind people. The results show that our plugin is capable of reducing the number of critical errors and that the majority is capable of identifying the improvements.

To sum up, this paper provides the following contributions:

- A catalog based on pattern matching to make websites more accessible;
- A tool implemented as a plugin to implement the catalog;
- A two-fold empirical study that demonstrate important and positive initial results, contributing to the blind people community that relies on screen readers.

2. Motivation

Digital accessibility is a legally guaranteed right for individuals with disabilities [Brazil 2000]. For example, in Brazil, the Brazilian Inclusion Law (LBI) establishes mandatory accessibility on all websites of public and private entities. This law focuses, for example, on blind people that depends on screen readers to navigate and interact online. The idea behind LBI is to guarantee equal access for people with disabilities [Brazil 2015]. Nevertheless, according to a study carried out by BigData Corp in partnership with the "Movimento Web para Todos" [All 2020], only 0.74% of the Brazilian websites meet the digital accessibility criteria. To better illustrate this scenario, we now introduce three examples from three websites containing problems to blind people that depends on screen readers.

The first one comes from the *Prefeitura Municipal de São Paulo*.[1] According to the Web Content Accessibility Guidelines (WCAG 2.1) Success Criterion 3.2.5 [Consortium 2018], when clicking on a link, blind people using screen readers must be informed that this link might take the user to a new window or tab. This is important because it helps the user to, for example, close the window or tab and get back to the origin page. To do so, it is possible to use the `target="_blank"` attribute along with an icon and an indication in the *title* attribute. This way, screen readers can inform what is written in the *title* attribute, making the user aware of the new page that will be opened. In our example, we show a website (Figure 1) that neither contains the `target="_blank"` attribute nor the *title* attribute.

Another problem for blind people is to be aware of the website contents when considering images. Screen readers can describe the visual content of the image by using the text written in the *alt* attribute. For people using screen readers, who cannot view the

[1] `https://www.capital.sp.gov.br/`

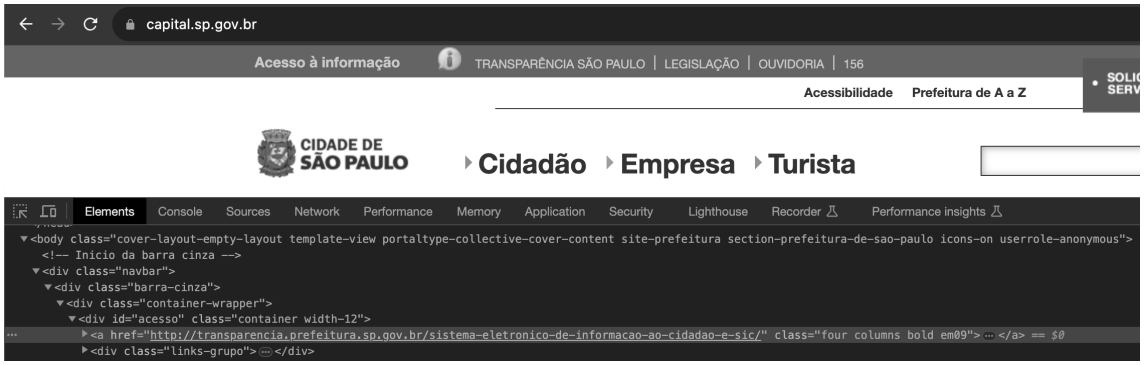


Figure 1. *Prefeitura Municipal de São Paulo* without the *target* and *title* attributes

images directly, the alternative text is read aloud, allowing them to understand the context and information conveyed by the image. This is particularly important in cases where the image contains relevant information or has a specific role in the content. In some cases, it may be appropriate for the *alt* attribute to be empty, such as purely decorative images that do not provide relevant information. Still, his presence should not be ignored. Figure 2 illustrates a bank website[2] that does not contain the *alt* attribute.

Figure 2. Website of a bank that does not contain the *alt* attribute

Last but not least, our third example is related to the we use the *lang* attribute (from the *html* tag) to specify the primary language of the page's content. This attribute plays an important role for screen readers, allowing this type of technology to present content appropriately to users. For example, it ensures that screen readers are able to correctly pronounce the words, in addition to language separation in cases where there are snippets of text in different languages. Thus, screen readers can identify and change the voice or language setting in each section, for a more comprehensible and coherent reading. Figure 3 illustrates an example from the *Portal da Transparência da Prefeitura de Maceió*.[3] In particular, this website does not contain the *lang* attribute.

3. A Preliminary Catalog to Make Websites More Accessible

In this section, we provide a preliminary catalog to make websites more accessible. Our catalog focuses on blind people that depend on screen readers when navigating on the internet. We present transformations in terms of a left-hand side and a right-hand side. Here, the left-hand side contains the problematic HTML code in terms of accessibility.

[2] `https://inter.co/`
[3] `https://www.transparencia.maceio.al.gov.br/`

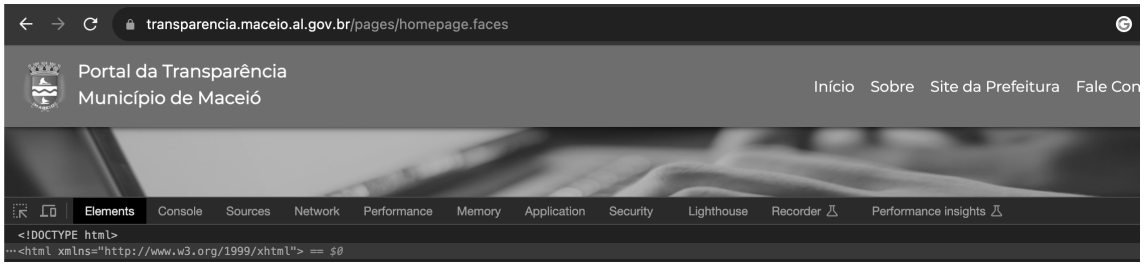


Figure 3. *Portal da Transparência da Prefeitura de Maceió* without the *lang* attribute

The right-hand side presents the HTML code "refactored", i.e., the HTML code that contains HTML tags and attributes that in general make the website more accessible to blind people that use screen readers.

Our catalog considers aspects such as defining the language of the document, including alternative text for images, handling new instances and modifying tables to include additional content, and elements to avoid the automatic opening of new windows or tabs without the user's request or awareness. The importance of this catalog lies in providing a clear and simple structure for including elements in an HTML page.

The catalog contains four transformations and rely on pattern matching. This means that, given a code that matches the left-hand side of one of our transformations, we convert the code to the corresponding right-hand side. Our catalog also relies on meta-variables. For example, *imgAttributes* is a meta-variable to represent the set of attributes of the *img* HTML tag. Another example: *languageAttribute* is a meta-variable to represent one specific attribute of the *html* tag, i.e., the *lang* attribute.

3.1. Alternative text for images

The left-hand side of this transformation consists of the *img* tag and the set of attributes of this tag, represented by the *imgAttributes* meta-variable. To apply the transformation to add the *alt* attribute (represented by the *altAttribute* meta-variable), the following precondition must be met: $altAttribute \notin imgAttributes$. The right-hand side contains the *img* tag with the *alt* attribute.

Transformation 1. Transformation to add the *alt* attribute

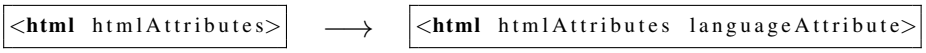
<**img** imgAttributes> $\longrightarrow$ <**img** imgAttributes altAttribute>

Precondition ($\longrightarrow$)
$altAttribute \notin imgAttributes$

3.2. Definition of the page language

Defining the language of the page using the *lang* attribute of the *html* tag is an important practice that guarantees the correct interpretation of the website contents by browsers, search engines and also by Assistive Technologies. In our catalog, at the left-hand side, we have the *html* tag along with the set of attributes of this tag (represented by the *htmlAttributes* meta-variable). To apply the transformation to add the *lang* attribute (represented by the *languageAttribute* meta-variable), the following precondition must be met: $languageAttribute \notin htmlAttributes$. The right-hand side contains the *html* tag with the *lang* attribute.

Transformation 2. Transformation to add the *lang* attribute



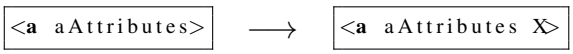
Precondition $(\longrightarrow)$
$languageAttribute \notin htmlAttributes$

3.3. Do not open new instances without the user request or awareness

The importance of not opening new instances without the user's request in the *a* tag is related to usability and the user's experience when browsing a website. When a link is configured to automatically open a new window or tab without the user intervention, it can result in a frustrating and confusing experience. Additionally, automatically opening new instances can cause reading and navigation confusion for users who rely on assistive technologies such as screen readers. These technologies present content in a linear fashion, and when new windows or tabs are opened without notice, it can make it difficult to understand the context and organize the content.

We provide a transformation in which the left-hand side contains the *a* tag and its set of attributes, here represented by the meta-variable *aAttributes*. To apply the transformation to add the *title* and the *target* attributes (represented by the *titleAttribute* and *targetAttribute* meta-variables, respectively), we have three preconditions and one of them must be met. In case $titleAttribute \notin aAttributes$ but $targetAttribute \in aAttributes$, we only add the *title* attribute. In case $titleAttribute \in aAttributes$ but $targetAttribute \notin aAttributes$, we only add the *target* attribute. Last but not least, we add both *title* and *target* attributes in case they are not in the set *aAttributes*. In particular, *targetAttribute* refers to `target="_blank"` and *titleAttribute* to the title of the target link. As mentioned, although it is not recommended to open links in new windows, in specific cases, it is possible to use `target="_blank"` together with an indication in the *title* attribute.

Transformation 3. Transformation to add the *title* and the *target* attributes



Precondition $(\longrightarrow)$
$titleAttribute \notin aAttributes \wedge targetAttribute \notin aAttributes \Rightarrow X = titleAttribute\ targetAttribute$
$titleAttribute \notin aAttributes \wedge targetAttribute \in aAttributes \Rightarrow X = titleAttribute$
$titleAttribute \in aAttributes \wedge targetAttribute \notin aAttributes \Rightarrow X = targetAttribute$

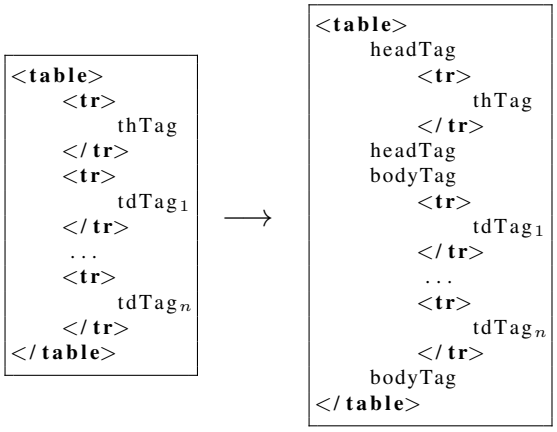
3.4. Add head and body information to better structure tables

When we customize the structure of HTML tables we can have important benefits for accessibility and understanding in the context of screen readers. These practices allow a more semantic organization of the table. In this context, the inclusion of the *thead* and *tbody* tags helps to make the table more accessible to screen readers, who are able to interpret and transmit information more clearly for blind people that depend on such equipments. By providing a clear structure and adequate contextual information, we improve the experience of using assistive technologies, ensuring a more efficient navigation and understanding of the tabulated contents.

To improve HTML tables and make them more accessible, we propose a transformation that relies on the *thead* and *tbody* tags (represented by the *headTag* and *bodyTag*

meta-variables). Moreover, we define a set of *th* tags (to define the table header) using the *thTag* meta-variable; and a set of *td* tags (to define the table columns), here represented by the *tdTag* meta-variable. At the left-hand side, the table must contain at least one *th* tag in the set *thTag*. Also, all the sets of the list [$tdTag_1$, $tdTag_2$, ..., $tdTag_n$] contains at least one *td* tag. The right-hand side yields a table containing the *thead* and *tbody* to better structure the table and bring benefits to the blind people community.

Transformation 4. Transformation to add the *thead* and *tbody* tags



Precondition ($\longrightarrow$)
$thTag$ contains at least one th tag
All the sets of the list $[tdTag_1, tdTag_2, ..., tdTag_n]$ contains at least one td tag

4. Implementing Our Preliminary Catalog

To automate our catalog, we introduce a plugin for the Visual Studio Code IDE. The main purpose of our plugin is to help developers fix accessibility issues in their web projects by analyzing and transforming the HTML source code automatically. To do so, our tool uses our preliminary catalog presented in the previous section. In particular, the plugin performs the following three steps:

1. The tool analyzes the code and searches for HTML code that matches any of the left-hand sides of our transformations;
2. All the left-hand sides collected in the previous step is stored in a list;
3. For each left-hand side in the list, the tool transforms the code based on the corresponding right-hand side.

The plugin also asks the programmer for additional information to be considered in the HTML code. For example, it asks for the language to be used in the *lang* attribute. It also asks for the text to describe images (*alt* attribute). For the *title* attribute of the *a* tag, we introduce a generic text informing that a new window or tab is about to be opened.

We name our plugin *a11y-refactoring* and it is available online.[4] Our plugin is open source so we invite the community to contribute and improve it. Our website contains information on how to install the plugin, how it works and presents details with respect to the transformations it performs.

[4] `https://github.com/easy-software-ufal/a11y-refactoring`

5. Evaluation

In this section, we evaluate our catalog and our tool to transform HTML code to make websites more accessible. Here, we intend to answer the following research questions:

- **RQ1:** To what extent our catalog makes websites more accessible when considering the standards from the W3C?
- **RQ2:** To what extent blind people perceive the improvements provided by our tool?

To answer these questions, we perform a two-fold empirical study. The first part relies on online tools capable of evaluating the quality of HTML code in terms of accessibility. The second part relies on survey with eight blind people.

5.1. Settings

To answer **RQ1**, we rely on three online evaluators: AChecker Web Accessibility, Access Monitor Plus, and WAVE Web Accessibility Evaluation Tools.[5] We select these evaluators because they follow the W3C standards and the WCAG guidelines. The steps to perform this evaluation are simple. First we submit to the evaluators the HTML code without our transformations and capture the metric *number of critical errors*, which is provided by these evaluators. Then, we use our plugin to perform improvements in the code. Finally, we submit the new HTML code version to check whether the online evaluators indicate improvements, i.e., whether the number of critical errors has decreased. The websites we used to answer **RQ1** are the same presented in Section 2: *Prefeitura Municipal de São Paulo*, *Banco Inter*, and *Portal da Transparência da Prefeitura de Maceió*. We only considered the main html file of these websites (i.e., *index.html*).

To answer **RQ2**, we recruit eight blind people to navigate in two different websites (one without the improvements of our catalog and another with the improvements). We surveyed the participants to check whether they perceived differences in both websites, even though we did not mention that one site contained problems and that the other contained our improvements. To execute our study, we implemented four simple websites with links, figures, tables, and other common HTML elements: *Pet shop with our improvements*; *Pet shop without our improvements*; *Bakery with our improvements*; and *Bakery without our improvements*.

We use the Latin Square Design [George E. P. Box and Hunter 2005] to structure our study. This way, we have two treatments: with and without our improvements. In this design, we dispose the blind people in rows and the websites in columns. The treatments come inside each cell. Notice that each treatment appears only once in every row and every column. Figure 4 presents our design. Notice that person 2 first executes the study in the website Bakery with our improvements; and then in the website Pet shop without our improvements. Person 1 executes in the opposite way: first without (in the Bakery) and then with the improvements (in the Pet shop).

This design is interesting to avoid learning effects. For example, if we let a person to navigate without and with our improvements in the same website (for example, Bakery), this subject would easily identify improvements by a simple comparison of the same website. This way, we would favor the second treatment (with our improvements).

[5] `https://achecks.org/achecker/`, `https://accessmonitor.acessibilidade.gov.pt/`, and `https://wave.webaim.org/`

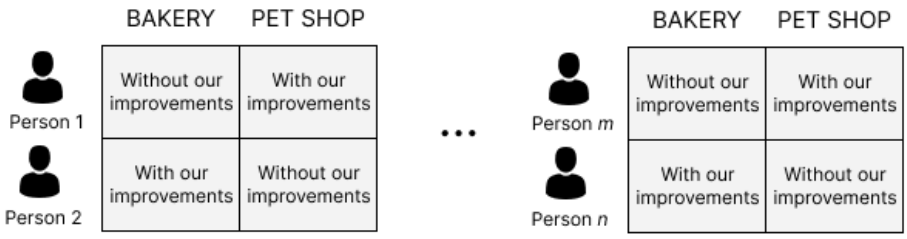


Figure 4. Latin Square design used in our study

The participants answered a Google Forms which asked what were their impressions in terms of accessibility when navigating in the first website and in the second. To better understand the participants backgrounds, we asked the screen reader technology they used and also their experience (Novice, Intermediate, or Advanced) in assistive technologies, such as the screen readers. To recruit the participants, we used online forms. To do so, we relied on blind people online communities that use screen readers. We provided a Consent Form and made clear that answers would be completely anonymous and that they could give up at any time.[6]

5.2. Results

We now present the results and answer our research questions. We have results of eight participants. All of them reported they are Intermediate or Advanced in using assistive technologies. Four used the Talkback screen reader, three used the NVDA, and one the VoiceOver. Figure 5 helps to answer **RQ1**. Notice that the number of critical errors has decreased in all three evaluators after submitting the websites with our improvements.

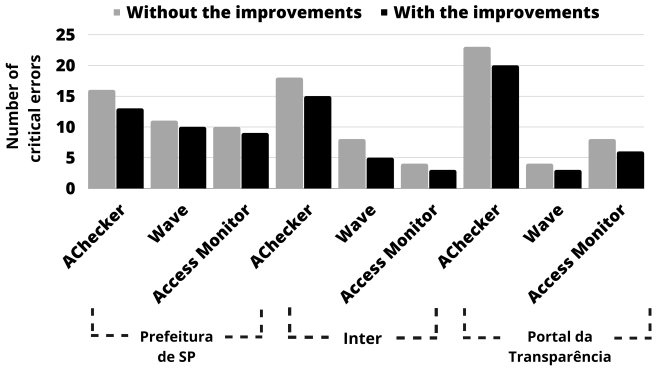


Figure 5. Number of critical errors with and without the improvements

Answer to **RQ1**: All evaluators pointed that the code improved by our plugin contained a slightly lower number of critical errors when compared to the original code.

[6]All artifacts used in our study are available at `https://github.com/easy-software-ufal/a11y-refactoring`

Regarding **RQ2**, seven (out of eight) participants perceived our improvements, mentioning terms directly related to the HTML tags and attributes that our plugin focuses. When the website contained the problems, the users were also capable of pointing them. We now present some quotes.

> "*There is no problem with the language and with the characters used in Portuguese, it has well-hierarchical headers and image descriptions.*"
>
> "*The links lead to the right places on the page, the titles correspond to the correct sections, the table at the end is well constructed.*"
>
> "*The language is not properly defined, causing errors in word accents. The images are not tagged with the alt attribute, making it impossible for a screen reader to tell which image is being viewed. The table is navigable.*"

Answer to **RQ2**: Seven (out of eight) participants have perceived the improvements of our plugin in two simple websites, although we did not mention that one site contained problems and that the other contained our improvements whatsoever. They even refer to elements from our catalog, such as the *lang* attribute, alternative text for images and tables navigable by screen readers, stating exactly the changes made by the plugin.

5.3. Threats to Validity

As threats to external validity, we cannot generalize our results. First, we used two simple websites we built. Second, we performed our study with only eight participants. Nevertheless, even with a small number of participants, they could identify the improvements, showing that simple changes in the HTML code can provide a huge benefit for blind people that use screen readers. As a threat to internal validity, we cannot guarantee that all participants were indeed blind, because our empirical study was online. We minimize this threat because we relied on online blind people communities.

6. Related Work

Previous works focused on improving navigability for users with low vision using screen magnification [Syed Masum Billah 2018]. However, although interaction with interfaces is widely researched in usability studies, these approaches primarily address visual accessibility on the web without fully considering the overall operability of a page's interface. Differently, our work focuses on HTML elements to improve the navigability. Similar to concerns faced by screen reader users, researchers also analyzed accessibility issues that affect the sequential order of user interface elements during navigation [Alshayban et al. 2020]. However, their study was limited to user interfaces on Android devices. Our work does not focus on mobile devices.

Previous researches delved into the effectiveness of image descriptions and the impact of their quality on the comprehension of web content by blind people [Rodríguez Vázquez 2016, Jeong et al. 2023, Calvo et al. 2016]. They emphasized the importance of providing accurate and informative alternative text for images to enhance web accessibility. This result is in accordance to ours, in which blind people cited the importance of the *alt* attribute.

7. Concluding Remarks

We introduced a preliminary catalog of transformations in HTML code to make websites more accessible. It relies on pattern matching (left-hand and right-hand side) and focuses on elements such as links, figures, and tables. We implement the catalog in terms of a plugin for the Visual Studio Code IDE. To evaluate our catalog and tool, we perform a two-fold empirical study. First, we check to what extent our improvements make websites more accessible by using online evaluators; then, we survey blind people to check if they perceive the improvements. All evaluators indicate a slightly lower number of critical errors after applying our plugin and seven out of eight of the blind participants was capable of identifying the improvements. They could even identify the exact improvements (e.g., a more structured table, the explicit definition of the website language). Our results reinforce the continued relevance of investing in accessibility improvements on the web, with special attention to the use of automated tools for correcting problems. As future work, we intend to improve our catalog, deal with server-side (e.g., PHP, Django) and client-side (e.g., Angular, React) frameworks and conduct more empirical studies.

References

All, W. M. F. (2020). Methodology used in the study of accessibility on active websites (brazil, 2020). Technical report.

Alshayban, A., Ahmed, I., and Malek, S. (2020). Accessibility issues in android apps: State of affairs, sentiments, and ways forward. In *Proceedings of the ICSE*, page 1323–1334. ACM.

Brazil (2000). Law no. 10,098, of december 19, 2000. *Diário Oficial da República Federativa do Brasil*.

Brazil (2015). Law no. 13,146, of july 6, 2015. *Diário Oficial da República Federativa do Brasil*.

Calvo, R., Seyedarabi, F., and Savva, A. (2016). Beyond web content accessibility guidelines: Expert accessibility reviews. In *Proceedings of the DSAI*, page 77–84. ACM.

Consortium, W. W. W. (2013). Web accessibility booklet - w3c. Technical report.

Consortium, W. W. W. (2018). Web content accessibility guidelines (wcag) 2.1. Technical report.

George E. P. Box, J. S. H. and Hunter, W. G. (2005). *Statistics for Experi- menters: design, innovation, and discovery*. Wiley-Interscience.

Jeong, H., Chun, M., Lee, H., Oh, S. Y., and Jung, H. (2023). Wataa: Web alternative text authoring assistant for improving web content accessibility. In *Companion Proceedings of IUI*, page 41–45. ACM.

Rodríguez Vázquez, S. (2016). Measuring the impact of automated evaluation tools on alternative text quality: A web translation study. In *Proceedings of the W4A*. ACM.

Syed Masum Billah, Vikas Ashok, D. E. P. I. R. (2018). Steeringwheel: A locality-preserving magnification interface for low vision web browsing. *ACM DL*, pages 3–9.

Systems, D. (2021). axe accessibility linter. [Accessed Aug 02, 2023].

van der Schee, M. (2018). Web accessibility. [Accessed Aug 02, 2023].