Trabalho de Conclusão de Curso

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

Ana Lúcia da Silva Ferreira
alsf@ic.ufal.br

Orientador:

Márcio de Medeiros Ribeiro

Maceió, March 2024

Ana Lúcia da Silva Ferreira

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

Monografia apresentada como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação do Instituto de Computação da Universidade Federal de Alagoas.

Orientador:

Márcio de Medeiros Ribeiro

Maceió, March 2024

Catalogação na Fonte
Universidade Federal de Alagoas
Biblioteca Central
Divisão de Tratamento Técnico
Bibliotecário: Marcelino de Carvalho Freitas Neto – CRB-4 - 1767

F383m Ferreira, Ana Lúcia da Silva.
Making websites more accessible for blind people with automatic HTML code transformations / Ana Lúcia da Silva Ferreira. – 2024.
39 f. : il.

Orientador: Márcio de Medeiros Ribeiro.
Monografia (Trabalho de conclusão de curso em Ciência da Computação) – Universidade Federal de Alagoas, Instituto de Computação. Maceió, 2024.

Bibliografia: f. 37-39.

1. WEB (Linguagem de programação) - Acessibilidade. 2. Experiência do usuário - Cegos. 3. Tecnologia de leitor de tela. 4. Ferramentas de acessibilidade para cegos. 5. Tecnologias assistivas. I. Título.

CDU: 004.43-056.262

Monografia apresentada como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação do Instituto de Computação da Universidade Federal de Alagoas, aprovada pela comissão examinadora que abaixo assina.

Márcio de Medeiros Ribeiro - Orientador
Universidade Federal de Alagoas

Elvys Alves Soares - Examinador
Centro de Informática
Universidade Federal de Pernambuco

Erick de Andrade Barboza - Examinador
Instituto de Computação
Universidade Federal de Alagoas

Maceió, March 2024

Agradecimentos

Aos que estiveram comigo desde antes do início, mas não puderam testemunhar o fim: meus pais, cuja gratidão pela presença e apoio transcendem quaisquer palavras.

Expresso minha profunda gratidão também aos professores do Instituto de Computação, especialmente ao meu orientador, Prof. Dr. Márcio de Medeiros Ribeiro, pelo tempo que trabalhamos juntos e pelo aprendizado adquirido.

A todos que contribuíram, direta ou indiretamente, para este trabalho, meu sincero agradecimento. Sua influência foi fundamental e valorizada. Este trabalho é fruto do apoio e esforço de muitos.

Resumo

A Lei Brasileira de Inclusão (LBI) exige acessibilidade em todos os websites, visando a inclusão digital, mas apenas 0,46% dos sites brasileiros atendem plenamente a esses requisitos. Apesar da eficácia das ferramentas atuais em identificar barreiras de acessibilidade, como a falta de descrições alternativas para imagens e informações de idioma, essenciais para usuários cegos que dependem de leitores de tela, elas não oferecem soluções automatizadas para corrigir tais problemas. Este trabalho propõe uma solução inovadora por meio de um catálogo preliminar e um plugin para o IDE Visual Studio Code, projetados para aprimorar a acessibilidade dos websites, detalhando transformações específicas no código HTML e facilitando sua implementação de maneira prática. A eficácia dessa abordagem foi validada em um estudo empírico que envolveu avaliações de acessibilidade online e a percepção de oito indivíduos cegos, mostrando uma redução significativa nos erros críticos de acessibilidade e confirmando o reconhecimento das melhorias pelos participantes. Este avanço representa um passo importante na direção de uma web mais inclusiva, oferecendo soluções práticas para superar as lacunas de acessibilidade e promover um acesso igualitário às informações e serviços digitais para todos os usuários, em especial para aqueles que dependem de leitores de tela.

Palavras-chave: Acessibilidade na Web, Experiência do Usuário de Indivíduos Cegos, Tecnologia de Leitor de Tela, Ferramentas de Acessibilidade para Cegos, Tecnologias Assistivas

Abstract

The Brazilian Inclusion Law (LBI) requires accessibility on all websites, aiming for digital inclusion, but only 0.46% of Brazilian websites fully meet these requirements. Despite the effectiveness of current tools in identifying accessibility barriers, such as the lack of alternative image descriptions and language information, essential for blind users who rely on screen readers, they do not offer automated solutions to correct such problems. This work proposes an innovative solution through a preliminary catalog and a plugin for the Visual Studio Code IDE, designed to improve the accessibility of websites, detailing specific transformations in the HTML code and facilitating their practical implementation. The effectiveness of this approach was validated in an empirical study that involved online accessibility assessments and the perception of eight blind individuals, showing a significant reduction in critical accessibility errors and confirming the participants' recognition of improvements. This advancement represents an important step towards a more inclusive web, offering practical solutions to overcome accessibility gaps and promote equal access to information and digital services for all users, especially those who depend on screen readers.

Key-words: Web Accessibility, User Experience of Blind Individuals, Screen Reader Technology, Accessibility Tools for the Blind, Assistive Technologies

List of Figures

Contents

1

Introduction

Web accessibility has become an increasingly discussed and relevant topic in the development of websites, since the digital inclusion of visually impaired or blind people is essential to guarantee their autonomy and access to information. The importance of transforming HTML code to improve website accessibility is widely recognized, especially when we consider the principles and techniques provided by the Web Content Accessibility Guidelines (WCAG) Consortium (2023a). These guidelines serve as a regulatory framework for creating more accessible digital content, highlighting the need to provide textual alternatives to images, ensure keyboard navigability, offer adaptable and understandable content presentations, and avoid elements that could cause seizures or reactions adverse physical conditions. In the realm of digital inclusivity, ensuring web accessibility for individuals with visual impairments, including those who are blind or rely on screen readers, is not merely a matter of compliance but a fundamental aspect of equitable access to information and services.

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.

The WCAG 2.2 guidelines, developed by the W3C, emphasize the need for web content to be perceivable, operable, understandable, and robust (POUR principles). These principles are crucial for ensuring that web content is accessible to all users, including those with disabilities. By adhering to these guidelines, developers can create web content that is perceivable through alternative texts for non-text content, operable via keyboard navigation, understandable with clear labels and instructions, and robust enough to be interpreted by a wide range of user agents, including assistive technologies. For visually impaired users and those who utilize screen readers, this translates into specific recommendations, such as providing alternative text for images (*alt* attributes), ensuring that all functionalities are accessible via keyboard, using semantic HTML tags for structure and navigation, and implementing Web Accessibility Initiative - Accessible Rich Internet Applications (WAI-ARIA) roles and properties to describe the behavior and purpose of dynamic content.

Moreover, the adoption of ARIA landmarks offers a way to navigate more efficiently through content, allowing users of assistive technologies to understand and interact with complex web applications. By embedding these practices into the web development process, developers not only adhere to legal and ethical standards but also enhance the user experience for a significant segment of the population that has been historically excluded in the digital space.

Legal mandates across the globe, such as the Americans with Disabilities Act (ADA) in the United States, the Accessibility for Ontarians with Disabilities Act (AOD) in Canada, and the European Accessibility Act (EAA) in the European Union, highlight the importance of web accessibility. These laws affirm the rights of individuals with disabilities to have equal access to information and services provided on the internet, mirroring the principles laid out in the United Nations Convention on the Rights of Persons with Disabilities (UNC).

In this context, the Brazilian Inclusion Law (LBI) reinforces the mandatory accessibility on all websites, positively impacting people who depend on assistive technologies, such as screen readers, to browse the internet. However, despite legislative advances and the availability of tools for diagnosing and correcting accessibility barriers, a study indicates that only a small fraction of Brazilian websites fully comply with accessibility standards. According to this study, conducted by BigDataCorp and referenced in Movimento Web Para Todos (2022), over 30 million Brazilian websites, both active and inactive, were analyzed, revealing that only a mere 0.46% of the 21 million active websites adhere to accessibility guidelines. These were specifically selected for analysis, excluding any that were offline or had not been updated for extended periods, with selection criteria considering the relevance of the website's content and the applicability of accessibility tests, taking into account the specificities of each type of website.

As shown in Table 1.1, the evolution of certain types of problems, such as checking the accessibility of form fields and buttons, links that open in a new window without prior warning and the absence of alternative texts for images shows a variation significant between April 2020 and May 2022. Additionally, checking HTML markup with W3C tools also indicated variability in site compliance, which reinforces the importance of adhering to web development best practices to ensure accessibility. This analysis highlights the ongoing need for improvements in web accessibility to meet established guidelines.

Types of problems	**April 2020**	**May 2021**	**May 2022**
Forms	55,19%	70,84%	53,46%
Links	93,65%	77,28%	87,30%
Images	83,36%	71,98%	84,21%
HTML Markup Check	97,22%	90,66%	95,78%

Table 1.1: Evolution of Accessibility Problems on Brazilian Websites.

Site Type (No Flaws)	2020	2021	2022
Blog	1,24%	2,17%	0,26%
Education	3,88%	4,68%	0,21%
E-commerce	1,30%	1,46%	0,06%
Corporate	2,81%	5,40%	0,16%
News portals	3,03%	3,15%	0,20%

Table 1.2: Percentage of Sites Without Failures in Accessibility Tests Per Type of Site.

The data in Tables 1.1 and 1.2 were derived from a comprehensive study by BigDataCorp, as referenced in Movimento Web Para Todos (2022).

This data highlights the importance of an ongoing commitment to web accessibility, not just as a matter of legal compliance, but as an essential element in ensuring that all users, regardless of their visual capabilities, can have equal access to information and services online. The evolution of website accessibility studies and the identification of conformity errors in HTML code markup are critical steps to move in this direction, highlighting the need for effective tools and greater awareness of the importance of digital accessibility.

This data highlights not only the variability in website accessibility over time, but also the persistent challenges related to HTML code markup compliance. Common errors continue to obstruct the browsing experience of users who rely on assistive technologies. As pointed out in Table 1.2 and Figure 1.1, the drop in the accessibility index in different types of websites between 2020 and 2022 presents the percentage of sites without failures in accessibility tests per site type, highlighting the areas that most require attention to ensure an inclusive user experience. It is imperative that developers, designers and website owners adopt tools that not only identify accessibility issues Systems (2021); van der Schee (2018), but that also promote the transformation of HTML code to eliminate these barriers, thus guaranteeing access more egalitarian and inclusive online information and services.

To minimize these problems, in this work we introduce a preliminary catalog of transformations in HTML code to make websites more accessible Ferreira and Ribeiro (2023). The catalog contains four transformations and rely on pattern matching. Each transformation is composed of 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.

The integration of accessibility features directly into the coding process, as advocated by our tool, represents a proactive approach to building a more inclusive web. This not only benefits individuals with visual impairments but also improves the overall quality and usability of web content for all users.

To evaluate our catalog and tool, we focus on two research questions: **RQ1:** To what extent

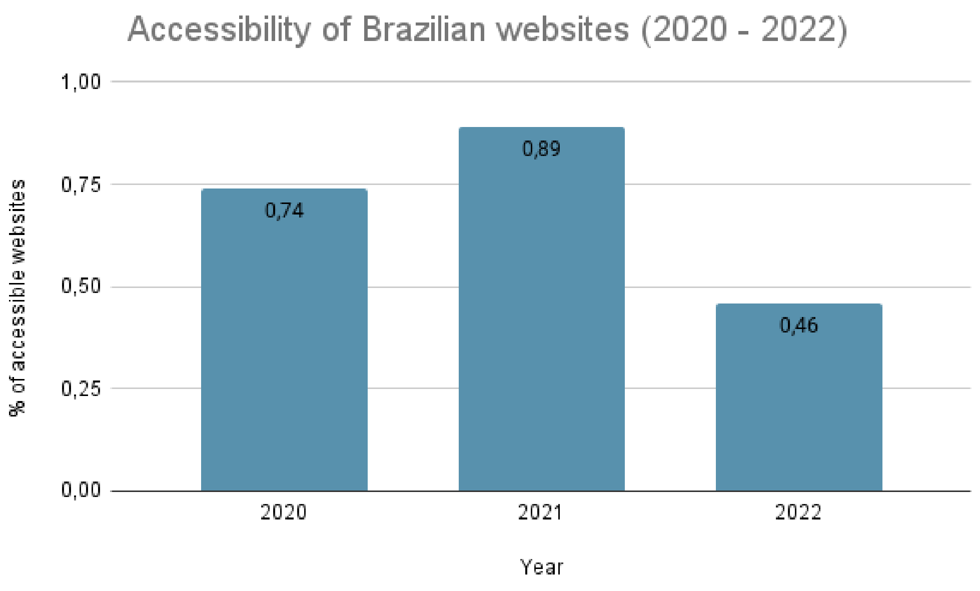


Figure 1.1: Accessibility of Brazilian websites (2020 - 2022).

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 with eight blind people, based on the Latin Square Design as described by Box et al. (2005). 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 work 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.

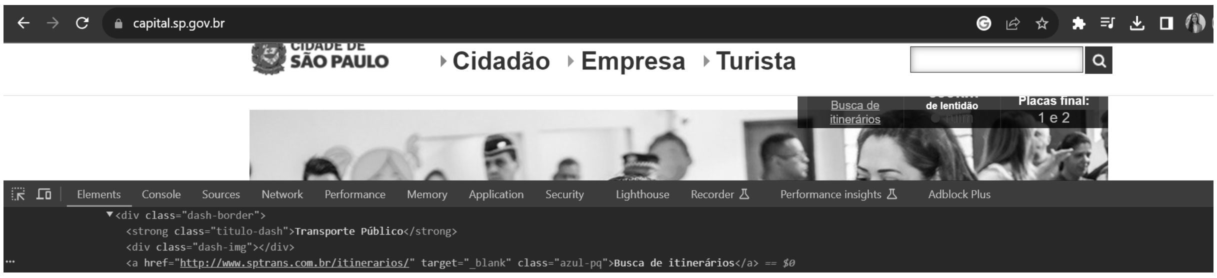
2 Motivation

According to Brazil (2000), digital accessibility is a legally guaranteed right for individuals with disabilities. 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 LBI, established by Brazil (2015), aims to guarantee equal access for people with disabilities. Nevertheless, according to a study carried out by BigData Corp in partnership with the " Movimento Web Para Todos (2022)", only 0.46% 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 2.2) Success Criterion 3.2.5 Consortium (2023b), 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 2.1) 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 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*

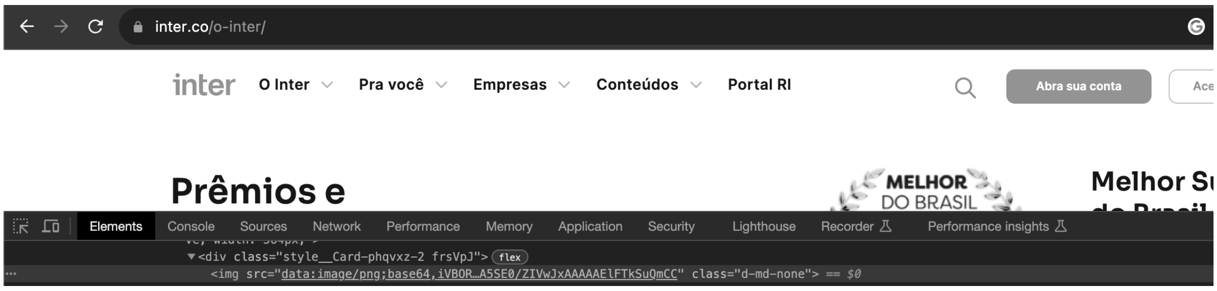
[1] https://www.capital.sp.gov.br/



```
▼<div class="dash-border">
   <strong class="titulo-dash">Transporte Público</strong>
   <div class="dash-img"></div>
   <a href="http://www.sptrans.com.br/itinerarios/" target="_blank" class="azul-pq">Busca de itinerários</a> == $0
```

Figure 2.1: *Prefeitura Municipal de São Paulo* with the *target* attribute and without *title* attribute.

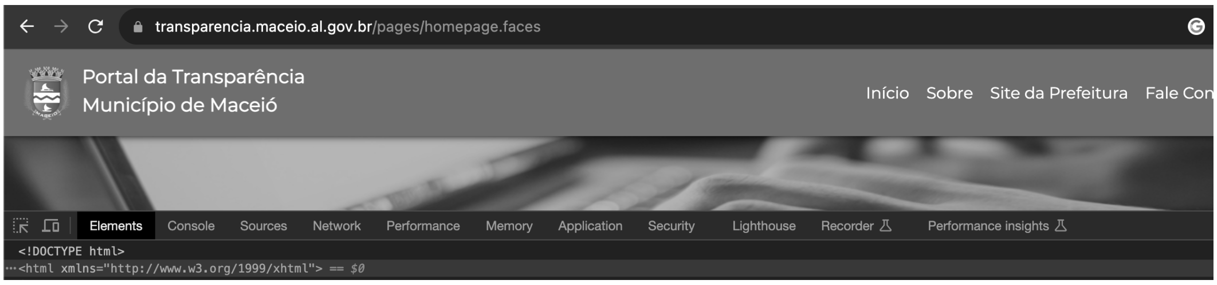
attribute to be empty, such as purely decorative images that do not provide relevant information. Still, his presence should not be ignored. Figure 2.2 illustrates a bank website[2] that does not contain the *alt* attribute.



```
▼<div class="style__Card-phqvxz-2 frsVpJ"> flex
   <img src="data:image/png;base64,iVBOR…A5SE0/ZIVwJxAAAAAElFTkSuQmCC" class="d-md-none"> == $0
```

Figure 2.2: Website of a bank that does not contain the *alt* attribute.

Additionally, we examine the use of 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 2.3 illustrates an example from the *Portal da Transparência da Prefeitura de Maceió*.[3] In particular, this website does not contain the *lang* attribute.



```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml"> == $0
```

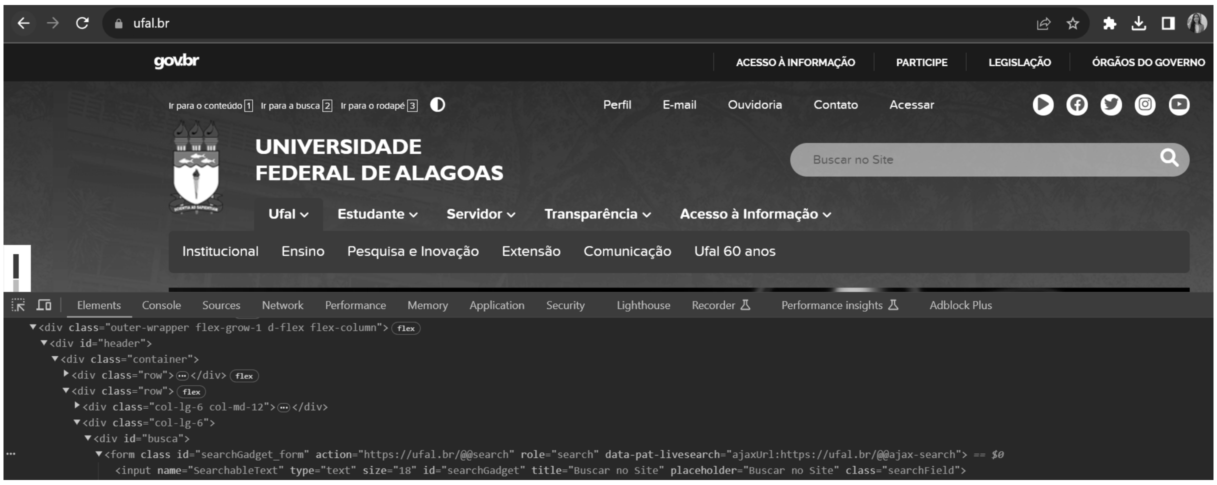
Figure 2.3: *Portal da Transparência da Prefeitura de Maceió* without the *lang* attribute.

Highlighting another crucial aspect of digital accessibility, we now turn our attention to the

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

significant role of *input* and *button* tags in the creation of accessible web interfaces. As defined by the Web Content Accessibility Guidelines (WCAG) 2.2, employing these tags effectively is essential to ensure that individuals with disabilities, particularly those who depend on assistive technologies, are able to efficiently interact with forms and other interactive controls on web pages.

In the case of the *input* tag, accessibility can be significantly improved through proper use of attributes like *aria-label* and *placeholder*. These attributes provide clear and concise textual descriptions, aiding users of screen readers to understand the purpose of each input field. For example, an input field for an email address should have a clear description that can be read by assistive technologies, thereby ensuring that all users comprehend its function and how to fill it out correctly. This practice aligns with WCAG 2.2 Success Criterion 1.1.1, which focuses on providing textual alternatives to non-textual content, and Criterion 3.3.2, requiring clear labels or instructions. Figure 2.4 illustrates an example of an input field enhanced without *aria-label*.

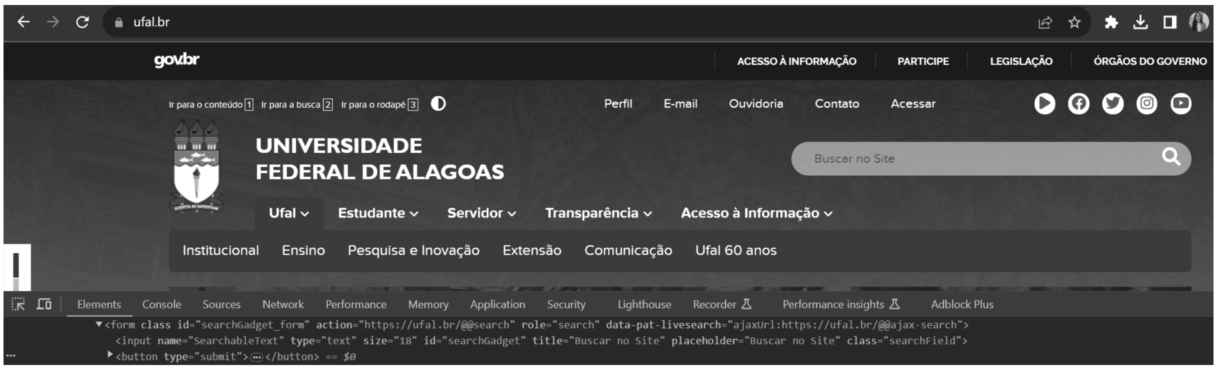


```
▼<div class="outer-wrapper flex-grow-1 d-flex flex-column"> flex
  ▼<div id="header">
    ▼<div class="container">
      ▶<div class="row"> ⋯ </div> flex
      ▼<div class="row"> flex
        ▶<div class="col-lg-6 col-md-12"> ⋯ </div>
        ▼<div class="col-lg-6">
          ▼<div id="busca">
            ▼<form class id="searchGadget_form" action="https://ufal.br/@@search" role="search" data-pat-livesearch="ajaxUrl:https://ufal.br/@@ajax-search"> == $0
              <input name="SearchableText" type="text" size="18" id="searchGadget" title="Buscar no Site" placeholder="Buscar no Site" class="searchField">
```

Figure 2.4: Input field on the Universidade Federal de Alagoas website without attribute *aria-label*.

Similarly, the *button* tag plays a vital role in navigation and interaction within a site. Clearly identified and described buttons not only facilitate the understanding of their purpose by users but also enable more intuitive interaction. The inclusion of the *aria-label* attribute in buttons, particularly those that use icons or images instead of text, is crucial to describe their function. This is particularly important for screen reader users, who may not be able to interpret the meaning of a visual icon. Adhering to WCAG 2.2, especially Success Criterion 4.1.2, ensures that each interface component's name, function, and value are understandable. Figure 2.5 demonstrates a button that is not accessible with an *aria-label* attribute.

Therefore, careful implementation of *input* and *button* tags, with special attention to adding accessible attributes, is an important step towards creating a more inclusive and accessible digital environment. By adhering to WCAG 2.2 guidelines, web developers can ensure that their content is accessible to a broader spectrum of users, including those with different types of disabilities.



```
▼<form class id="searchGadget_form" action="https://ufal.br/@@search" role="search" data-pat-livesearch="ajaxUrl:https://ufal.br/@@ajax-search">
   <input name="SearchableText" type="text" size="18" id="searchGadget" title="Buscar no Site" placeholder="Buscar no Site" class="searchField">
  ▶<button type="submit">⋯</button> == $0
```

Figure 2.5: Button on the Universidade Federal de Alagoas website without attribute *aria-label*.

3

A Preliminary Catalog to Make Websites More Accessible

In this chapter, 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. 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 eight 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 3.1: Transformation to add the *alt* attribute

$$\boxed{\textbf{<img} \ \texttt{imgAttributes>}} \longrightarrow \boxed{\textbf{<img} \ \texttt{imgAttributes altAttribute>}}$$

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

```
<div id="header">
  <div class="row-top">
    <div class="row-content">
      <div class="logo">
        <a href="/portal/home.action"
           class="logo"
           title="Ir para a p gina inicial do e-Aulas">
          <img src="/portal/skins/default/images/logo_usp.png">
        </a>
      </div>
    </div>
  </div>
</div>
```

Listing 3.1: Portal de Aulas da USP's code without the *alt* (left-hand side) attribute

```
<main id="header">
  <div class="row-top">
    <div class="row-content">
      <div class="logo">
        <a href="/portal/home.action"
           class="logo"
           title="Ir para a p gina inicial do e-Aulas">
          <img src="/portal/skins/default/images/logo_usp.png" alt="">
        </a>
      </div>
    </div>
  </div>
</main>
```

Listing 3.2: Portal de Aulas da USP's code with the *alt* (right-hand side) attribute

In this context, Listing 3.1 and Listing 3.2 present real cases that exemplify the need and implementation of one of the transformations proposed in the catalog to make websites more accessible.

Listing 3.1 shows the original code from the Portal de Aulas da USP[1] without the *alt* attribute, illustrating the left-hand side of the transformation. Listing 3.2 shows the same code after adding the *alt* attribute, representing the right-hand side of the transformation.

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 3.2: Transformation to add the *lang* attribute

```
<html htmlAttributes>
```

$\longrightarrow$

```
<html htmlAttributes languageAttribute>
```

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

In this context, let's present two real cases exemplifying the application of this transformation.

```
<!-- saved from url=(0033)https://eaulas.usp.br/portal/home -->
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=
           UTF-8" />
        <meta name="color-scheme" content="light dark" />
    </head>
    <body>
        <div class="line-gutter-backdrop"></div>
        <form autocomplete="off">
            <label class="line-wrap-control">Moldar linhas<input type
               ="checkbox" aria-label="Moldar linhas" /></label>
        </form>
    </body>
</html>
```

Listing 3.3: A website's code without the *lang* attribute (left-hand side)

[1]https://eaulas.usp.br/portal/home

```html
<!-- saved from url=(0033)https://eaulas.usp.br/portal/home -->
<html lang="pt-br">
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=
           UTF-8" />
        <meta name="color-scheme" content="light dark" />
        <meta charset="UTF-8" />
    </head>
    <body>
        <main class="line-gutter-backdrop"></main>
        <form autocomplete="off">
            <label class="line-wrap-control">Moldar linhas<input type
               ="checkbox" aria-label="Moldar linhas" /></label>
        </form>
    </body>
</html>
```

Listing 3.4: The same website's code with the *lang* attribute added (right-hand side)

Listing 3.3 shows the original code from the Portal de Aulas da USP without the *lang* attribute, illustrating the left-hand side of the transformation, while Listing 3.4 displays the code after adding the *lang* attribute, representing the right-hand side. These listings serve as practical examples of how adding the *lang* attribute can significantly enhance a website's accessibility for users who rely on assistive technologies, as well as for search engines.

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

Understanding the need for clarity in website navigation, especially for users who rely on assistive technologies such as screen readers, WCAG 2.2 establishes guidelines that are essential when configuring links to open in new windows or tabs. The approach of adding a descriptive title to links that open in new instances is crucial for accessibility and usability, as it avoids confusion and improves the user experience.

WCAG 2.2 success criteria, such as those that focus on the understandability of the purpose of links, both in context and in isolation, are particularly relevant here. They ensure that users, especially those using screen readers, can understand what to expect from each link. For example, when a screen reader user navigates a page, the software reads the available information about each element. If a link is configured to open in a new window but does not have an explanatory title, the user may not be aware of this functionality. This can lead to a disorienting and frustrating browsing experience, especially if the user doesn't realize that a new window has opened.

Screen readers work by presenting the content of a page in a linear and sequential manner. When a link opens a new window without warning, the user may suddenly find themselves in a new context without a clear understanding of how they got there. This can be particularly problematic if the user tries to return to the previous page and is unable to find their way back, resulting in a frustrating and confusing experience.

The WCAG 2.2 Success Criteria addresses these challenges by providing specific guidance on how links should be structured to improve accessibility. For example, by adding a descriptive title to a link that opens in a new window, developers can clearly inform users about the link's behavior. This is in line with Success Criterion 2.4.4, which focuses on understandability of the purpose of links in context, and Success Criterion 2.4.9, which extends understandability of link purpose even when isolated from its context.

These criteria are fundamental because they ensure that users, regardless of their skills or the technologies they use, have enough information to make informed decisions when browsing. For example, a clear and descriptive title on a link not only benefits screen reader users, but also all users by providing a clear understanding of what to expect when interacting with the link.

Additionally, Success Criterion 4.1.2, which addresses the need for user interface components to convey information about their name, function, and value, emphasizes the importance of clearly communicating the nature of links. This includes informing whether the link will open in a new window, a piece of information that is vital for the user's conscious and controlled navigation.

In summary, implementing best practices as outlined by WCAG 2.2 not only helps comply with legal and ethical accessibility standards, but also improves the overall user experience. By providing clear and predictable information about links and their behavior, developers can create a more intuitive and accessible browsing environment, promoting a more inclusive web for everyone.

Transformation 3.3: Transformation to add the *title* attribute

$$\boxed{\texttt{<a aAttributes>}} \longrightarrow \boxed{\texttt{<a aAttributes X>}}$$

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

In the context of enhancing web accessibility, Listing 3.5 and Listing 3.6 present real cases illustrating the application of this transformation.

```
<a
  class="html-attribute-value html-external-link"
  target="_blank"
  href="https://g1.globo.com/bom-dia-brasil/noticia/conheca-a-
     historia-do-bom-dia-brasil-desde-a-estreia.ghtml"
  rel="noreferrer noopener"
>
  https://g1.globo.com/bom-dia-brasil/noticia/conheca-a-historia-do-
     bom-dia-brasil-desde-a-estreia.ghtml
</a>
```

Listing 3.5: A link without a descriptive title, lacking clarity on the action of opening in a new window

```
<a
  class="html-attribute-value html-external-link"
  target="_blank"
  href="https://g1.globo.com/bom-dia-brasil/noticia/conheca-a-
     historia-do-bom-dia-brasil-desde-a-estreia.ghtml"
  rel="noreferrer noopener"
  title="Link to external site (opens in a new window)"
>
  https://g1.globo.com/bom-dia-brasil/noticia/conheca-a-historia-do-
     bom-dia-brasil-desde-a-estreia.ghtml
</a>
```

Listing 3.6: The same link with a descriptive title added, clearly indicating that it opens in a new window

Listing 3.5 and Listing 3.6, sourced from the G1 Globo website[2], compellingly underscore the importance of providing clear, detailed descriptions for links, especially those that open in new windows or tabs. They highlight a prevalent issue where the absence of information on the link's behavior can lead to confusion among users. By adhering to the WCAG 2.2 guidelines and incorporating modifications such as descriptive titles, developers can significantly enhance website accessibility. This proactive approach ensures that all users, particularly those dependent on assistive technologies, are well-informed about the outcomes of their interactions. Such a commitment to clarity and predictability is fundamental in creating a digital environment that values and facilitates every user's experience, thereby making the web a more inclusive space for everyone.

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

[2] https://g1.globo.com/

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. In instances where *thTag* is absent, *headTag* will encapsulate the first block of *tr* tags, subtly addressing situations that lack explicit header rows. The right-hand side yields a table containing the *thead* and *tbody* to better structure the table and bring benefits to the blind community.

Transformation 3.4: Transformation to add the *thead* and *tbody* tags

```
<table>
    <tr>
        thTag
    </tr>
    <tr>
        tdTag_1
    </tr>
    ...
    <tr>
        tdTag_n
    </tr>
</table>
```

$\longrightarrow$

```
<table>
    headTag
        <tr>
            thTag
        </tr>
    headTag
    bodyTag
        <tr>
            tdTag_1
        </tr>
        ...
        <tr>
            tdTag_n
        </tr>
    bodyTag
</table>
```

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

```
<div id="afterGraph"></div>
<table id="table_dolar_bitcoin">
  <tbody>
    <tr>
      <td><a href="/dolar">D LAR</a></td>
      <td class="negative">-0,72%</td>
      <td>R$ 4,96</td>
    </tr>
    <tr>
      <td><a href="/bitcoin">BITCOIN</a></td>
      <td class="negative">-1,91%</td>
      <td>R$ 24393,00</td>
    </tr>
  </tbody>
</table>
```

Listing 3.7: Example table from the InfoMoney website before transformation. This table, although it contains the *tbody* tag, does not have the *thead* tag, limiting accessibility and semantic organization for screen reader users

```
<div id="afterGraph"></div>
<table id="table_dolar_bitcoin">
  <thead>
    <tr>
      <td><a href="/dolar">D LAR</a></td>
      <td class="negative">-0,72%</td>
      <td>R$ 4,96</td>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td><a href="/bitcoin">BITCOIN</a></td>
      <td class="negative">-1,91%</td>
      <td>R$ 24393,00</td>
    </tr>
  </tbody>
</table>
```

Listing 3.8: Example table from the InfoMoney website after transformation, with the inclusion of *thead* and *tbody* tags

The examples in Listing 3.7 and Listing 3.8, taken directly from the InfoMoney website, illustrate the importance of properly structuring HTML tables for accessibility. Listing 3.7 shows a table before applying the proposed transformation, where the absence of the *thead* tag prevents efficient reading by assistive technologies. In contrast, Listing 3.8 highlights the result after including the *thead* and *tbody* tags, demonstrating a notable improvement in the semantic organization and accessibility of the table. These changes not only make navigation easier for users who rely on screen readers, but also reinforce the importance of following web development best practices as suggested by the WCAG accessibility guidelines.

3.5 Improving Accessibility in Forms with Aria-Label Attributes

When addressing the accessibility of input fields and text areas, the distinction between the use of *placeholders* and the application of *aria-labels* or associated visual tags (labels) proves crucial, especially for users who rely on assistive technologies, such as screen readers. While *placeholders* offer a temporary visual cue about the type of information required in a field, such as "Enter your name," they have significant limitations in terms of accessibility. These texts disappear as soon as typing begins, which can confuse users who need continuous visual reminders or who use screen readers, who may not consistently pick up on such temporary cues.

On the other hand, the *aria-label* attribute plays a key role in providing a permanent and accessible textual description for input fields, compensating for the absence of visual labels. This practice not only makes it easier for all users, including those using assistive technologies, to understand the purpose of each field, but also meets important success criteria of Web Content Accessibility Guidelines (WCAG) 2.2.

Specifically, the insertion of aria-labels or the association with clear visible labels, even in the presence of placeholders, is recommended to ensure better accessibility and usability. This aligns with several WCAG success criteria, including the need to offer textual alternatives to non-textual content (Criterion 1.1.1), ensure clear labels and instructions for navigating and understanding forms (Criteria 2.4.6 and 3.3. 2), and ensure that the name, function and value of each interface component are understandable (Criterion 4.1.2).

Therefore, the adoption of *aria-labels*, in addition to or instead of *placeholders*, significantly contributes to the creation of more inclusive digital experiences. This approach not only improves accessibility for users who are visually impaired or who rely on screen readers, but also enhances usability for all users by ensuring that the functions and purposes of input fields are understood clearly and consistently, regardless of how they are accessed or perceived.

Transformation 3.5: Transformation to add the *aria-label* attribute

$$\boxed{\texttt{<input inputAttributes>}} \longrightarrow$$

$$\boxed{\texttt{<input inputAttributes ariaLabelAttribute>}}$$

Precondition $(\longrightarrow)$

$ariaLabelAttributes \notin inputAttributes \wedge (nameAttribute \in inputAttributes \vee idAttribute \in inputAttributes \vee placeholderAttribute \in inputAttributes) \Rightarrow ariaLabelAttribute = \text{value from } nameAttribute, idAttribute, \text{ or } placeholderAttribute$

The specified precondition indicates conditional logic for adding an *aria-label* attribute to

`<input>` tag in an HTML document. It states that, for an `<input>` element that does not currently have an *aria-label* attribute, an *aria-label* will be added if the element has a *name*, *id*, or *placeholder* attribute. The value assigned to the *aria-label* will be derived from the value of the *name*, *id*, or *placeholder* attribute available on the element. This condition is designed to ensure that all input elements have an accessible description, improving usability for users who rely on assistive technologies such as screen readers. This transformation process is visualized in the listings below, where a `<input>` element with no aria-label is transformed to include one, with the appropriate value derived as specified.

To further illuminate the discussion on enhancing form accessibility through the use of *aria-label* attributes, let's consider practical examples derived from the G1 Globo website. These instances vividly demonstrate the transformation from a less accessible form input to one optimized for accessibility, highlighting the critical distinctions and improvements made.

```
<input
  placeholder="BUSCAR"
  type="search"
  name="q"
  id="busca-campo"
  autocomplete="off"
  tabindex="1"
/>
```

Listing 3.9: Example of a form input from the G1 Globo site before applying the *aria-label* attribute. This input relies solely on a placeholder for user guidance, which, while visually informative, falls short in accessibility for users utilizing screen readers

```
<input
  placeholder="BUSCAR"
  type="search"
  name="q"
  id="busca-campo"
  autocomplete="off"
  tabindex="1"
  aria-label="BUSCAR"
/>
```

Listing 3.10: The same form input from the G1 Globo site after the inclusion of the *aria-label* attribute. This modification enhances the input's accessibility by providing a persistent, screen-reader-friendly description, ensuring that all users understand its purpose

The contrast between Listings 3.9 and 3.10 underscores the pivotal role of the *aria-label* attribute in bridging the accessibility gap present in form inputs. Initially, the reliance on placeholders (Listing 3.9) presents a significant barrier to accessibility, as these cues vanish once interaction begins, potentially disorienting users who depend on assistive technologies. The subsequent transformation (Listing 3.10), achieved through the strategic application of *aria-labels*, exemplifies a commitment to inclusive design. By providing a permanent, audible description for input fields, the modification directly addresses the limitations highlighted in the initial setup and aligns with the best practices recommended by WCAG 2.2 for creating universally accessible digital environments. These examples from the G1 Globo website serve not only as a testament to the efficacy of adopting *aria-labels* in web forms but also as a guide for developers seeking to enhance the usability and accessibility of their websites for all users, particularly those utilizing screen readers.

3.6 Improving Web Accessibility with Aria Labels in Interactive Elements

In the context of digital accessibility, particularly in adherence to the Web Content Accessibility Guidelines (WCAG) 2.2, the implementation of textual descriptions for interactive elements, such as buttons and links acting as buttons, is essential. This method ensures a degree of clarity and context often absent in user interface designs, proving critical for users dependent on assistive technologies like screen readers.

Applying textual descriptions to elements that lack distinct visible text enhances accessibility for individuals with visual impairments. For instance, elements solely utilizing icons or symbols become significantly more accessible when paired with comprehensible textual labels. Differing from *data-label* attributes, which serve coding or styling purposes and aren't inherently accessible to screen readers, *aria-label* attributes are crafted to bolster accessibility. Aria-labels delineate the function and intent of an element, enabling screen readers to interpret and relay this information to users, thereby bridging a crucial gap in the user interface for those with visual impairments.

This tactic aligns with WCAG Success Criterion 1.1.1, which is about providing textual alternatives for non-textual content. For links that double as buttons, concise and descriptive *aria-label* clarify their function, making it more intuitive and accessible. This complies with Success Criterion 2.4.4, underscoring the importance of understanding the purpose of each link within its context.

Moreover, by meeting Success Criterion 4.1.2, every user interface component is made identifiable and operable through assistive technologies, ensuring that users of screen readers can comprehend and interact effectively with the website.

Hence, the incorporation of *aria-labels* into interactive elements, in contrast to *data-labels*, presents a potent measure to enhance accessibility. This facilitates a more efficient and inclusive navigation and interaction with online content for people of diverse abilities.

The specified precondition describes the logic for adding an *aria-label* attribute to `<button>`

Transformation 3.6: Transformation to add the *aria-label* attribute

```
<button buttonAttributes>
```
$\longrightarrow$
```
<button buttonAttributes ariaLabelAttribute>
```

Precondition ($\longrightarrow$)
$ariaLabelAttribute \notin buttonAttributes \wedge$ buttonContent is empty $\wedge$
$(dataLabelAttribute \in buttonAttributes \vee titleAttribute \in buttonAttributes \vee idAttribute \in buttonAttributes)$
$\Rightarrow ariaLabelAttribute =$ value from $dataLabelAttribute, titleAttribute,$ or transformed $idAttribute$

tags in the context of digital accessibility, under specific conditions. This transformation is only applied when the button does not initially have an *aria-label*, does not contain visible text, and has at least one of the following attributes: *data-label*, *title*, or *id*. The logic determines that the value for the *aria-label* must be derived from one of these three attributes, prioritizing the *data-label*, followed by the *title*, and, if none of the previous ones are available, transforming the *id* value into a more readable form for use as *aria-label*. This approach aims to improve the accessibility of interactive elements for users of assistive technologies, such as screen readers, by providing clear text provision for actions represented by icons or non-text visual elements.

To further illustrate the importance of implementing aria labels for enhancing web accessibility, let's examine examples from the Receita Federal website[3]. These examples demonstrate the application of aria labels to interactive elements, such as buttons that lack visible text but play a crucial role in user navigation and interaction.

```
<button class="aa-SubmitButton" type="submit" title="Submit">
  <svg class="aa-SubmitIcon" viewBox="0 0 24 24" width="20"
     height="20" fill="currentColor">
    <path d="M16.041 15.856c-0.034 0.026-0.067 0.055-0.099
       0.087s-0.060 0.064-0.087 0.099c-1.258"/>
  </svg>
</button>
```

Listing 3.11: An example of a button from the Receita Federal site before the addition of an *aria-label*

```
<button class="aa-SubmitButton" type="submit" title="Submit"
    aria-label="Submit">
  <svg class="aa-SubmitIcon" viewBox="0 0 24 24" width="20"
     height="20" fill="currentColor">
    <path d="M16.041 15.856c-0.034 0.026-0.067 0.055-0.099
       0.087s-0.060 0.064-0.087 0.099c-1.258"/>
  </svg>
```

[3] https://www.gov.br/receitafederal/pt-br

```
</button>
```

Listing 3.12: The same button from the Receita Federal site after implementing an *aria-label* attribute. This modification significantly improves the button's accessibility by providing a clear, textual description of its function, thus making it accessible to and operable by users of assistive technologies

The contrast between Listings 3.11 and 3.12 highlights the transformative impact of aria labels on the accessibility of web content. Initially, the button's function (Listing 3.11) might be unclear to users utilizing screen readers due to the absence of a descriptive label. The introduction of an *aria-label* (Listing 3.12), as seen on the Receita Federal website, exemplifies a best practice in digital accessibility. By providing a permanent, audible cue that clearly describes the button's action, this approach addresses a critical accessibility gap, ensuring that all users, especially those with visual impairments, can navigate and interact with the website more effectively.

These examples from the Receita Federal website serve not only as a testament to the practical benefits of aria labels in web development but also as a compelling call to action for designers and developers to prioritize accessibility. Incorporating *aria-labels* into interactive elements significantly enhances the user experience for a broad audience, reinforcing the principle that web accessibility is a cornerstone of inclusive design and a necessity for creating equitable digital spaces.

3.7 Changing Generic Divs for Semantic Structures in HTML

In the context of digital accessibility, particularly under the WCAG 2.2 guidelines, the adoption of a correct semantic structure in HTML documents is of paramount importance. The practice of replacing generic elements like <*div*> with specific semantic elements like <*header*>, <*nav*>, and <*main*> plays a crucial role in this context. This approach not only improves the structural clarity of the page, but also makes the content more accessible, especially for users who rely on assistive technologies such as screen readers.

By employing appropriate semantic elements, the relationship between the page's visual structure and its programmatic representation becomes clearer and more direct. This is in line with WCAG Success Criterion 1.3.1, which emphasizes the importance of information and relationships being programmatically understandable. For example, by replacing a <*div*> with a <*header*>, it provides clear context about the nature of that section of the page, improving understanding and navigation.

Additionally, this approach makes it easier to implement skipping mechanisms, aligning with Success Criterion 2.4.1, which focuses on the ability to skip blocks of content. Elements like <*nav*> allow screen reader users to quickly navigate menus without having to scroll through content linearly.

Using these semantic elements not only meets the accessibility criteria established by

WCAG, but also enhances the overall user experience. It makes it easier for all users to understand the structure and layout of the page, contributing to more intuitive and efficient navigation.

To underscore the significance of adopting semantic HTML structures for enhancing digital accessibility, let's consider transformative examples from the Receita Federal website. These instances showcase the transition from using generic <*div*> elements to incorporating specific semantic tags like <*header*>, <*nav*>, and <*main*>, highlighting the substantial benefits this change brings to users, particularly those utilizing assistive technologies.

```
<div class="links-rapidos">
  <a class="toggle-links-rapidos" href="#">
    <span class="fas fa-ellipsis-v"></span>
    <span class="sr-only">Acesso rápido</span>
  </a>
  <ul>
    <li class="titulo">Acesso rápido</li>
    <li>
      <a href="https://www.gov.br/pt-br/orgaos-do-governo">
          rg ãos do Governo</a>
    </li>
    <li>
      <a href="https://www.gov.br/acessoainformacao/pt-br">
         Acesso à Informação</a>
    </li>
    <li>
      <a href="http://www4.planalto.gov.br/legislacao">
         Legislação</a>
    </li>
    <li>
      <a href="https://www.gov.br/governodigital/pt-br/
         acessibilidade-digital">Acessibilidade</a>
    </li>
  </ul>
</div>
```

Listing 3.13: A section of the Receita Federal website before the semantic enhancement, utilizing generic <*div*> elements

```
<nav class="links-rapidos">
  <a class="toggle-links-rapidos" href="#">
    <span class="fas fa-ellipsis-v"></span>
    <span class="sr-only">Acesso rápido</span>
  </a>
  <ul>
    <li class="titulo">Acesso rápido</li>
    <li>
      <a href="https://www.gov.br/pt-br/orgaos-do-governo">
          rg ãos do Governo</a>
    </li>
    <li>
      <a href="https://www.gov.br/acessoainformacao/pt-br">
         Acesso à Informação</a>
    </li>
    <li>
      <a href="http://www4.planalto.gov.br/legislacao">
         Legislação</a>
    </li>
    <li>
      <a href="https://www.gov.br/governodigital/pt-br/
         acessibilidade-digital">Acessibilidade</a>
    </li>
  </ul>
</nav>
```

Listing 3.14: The same section of the Receita Federal website after incorporating semantic HTML elements

The juxtaposition of Listings 3.13 and 3.14 illustrates the transformative effect of implementing semantic HTML on web accessibility. Initially, the reliance on <*div*> elements (Listing 3.13) presents an accessibility challenge, as it fails to convey the structural significance of different page sections to screen reader users. The evolution towards a semantic structure (Listing 3.14), as demonstrated on the Receita Federal website, exemplifies best practices in web development. This change not only facilitates improved navigation and comprehension for users of assistive technologies but also aligns with WCAG 2.2 guidelines by making the page's structure programmatically understandable.

These examples from the Receita Federal website serve as a compelling testament to the practical and inclusive benefits of semantic HTML. By emphasizing the structural and functional roles of different page sections through specific HTML tags, developers can significantly enhance the user experience for a diverse audience, ensuring that the web remains an accessible and navigable space for everyone. This approach not only meets key WCAG criteria but also fosters a more inclusive digital environment, reinforcing the importance of accessibility in modern web design.

3.8 Improving the Accessibility of Iframes with Title Attributes

Within the scope of digital accessibility, especially as outlined by WCAG 2.2, the practice of ensuring that <*iframe*> elements have title attributes is a crucial step in making web content more accessible. This approach serves to provide context and descriptions to elements that might otherwise be ambiguous or inaccessible to users of assistive technologies such as screen readers.

Including titles in <*iframe*> significantly improves the user experience for people with visual impairments. These titles help communicate the content or purpose of the <*iframe*>, especially when the content is external or not immediately apparent. This aligns with the need, as expressed in WCAG, that all information, structures and relationships presented visually must also be accessible through programmatic means, allowing users of assistive technologies to understand and navigate content effectively.

Additionally, by making it easier to implement titles, even if they are initially empty, developers and publishers are encouraged to provide more descriptive and meaningful titles. This not only meets WCAG's success criteria of allowing users to skip blocks of content and understand the function and value of interactive elements, but also improves the overall accessibility of the site.

This practice of adding titles to <*iframe*>, although it may start with an empty default value, highlights the importance of making all page elements accessible and understandable. Thus, it significantly contributes to a more inclusive and enriching browsing experience for all users, especially those who depend on assistive technologies.

Transformation 3.7: Transformation to add the *title* attribute

`<iframe iframeAttributes>` $\longrightarrow$ `<iframe iframeAttributes X>`

Precondition $(\longrightarrow)$
$titleAttribute \notin iframeAttributes \Rightarrow X = titleAttribute$

To further elucidate the enhancement of *iframe* accessibility through the addition of *title* attribute, let's examine practical examples from the ClimaTempo website[4]. These instances vividly illustrate the before and after of applying title attributes to iframes, showcasing the tangible benefits of such modifications for users relying on assistive technologies.

```
<div id="tbl-aug-8d9y1q" class="trc_popover_aug_container">
  <div id="tbl-aug-2e8dxn" class="trc_popover_aug_container">
    <div id="tbl-aug-h7cua9" class="trc_popover_aug_container
        ">
```

[4] https://www.climatempo.com.br/

```
      <div class="trc_popover trc_popover_fade trc_bottom">
        <div class="trc_popover_arrow"></div>
        <iframe frameborder="0" scrolling="no" src="
           javascript:void(0)" style="width: 100%">
        </iframe>
      </div>
    </div>
  </div>
</div>
```

Listing 3.15: Example of an <*iframe*> from the ClimaTempo site with the *title* attribute absent

```
<div id="tbl-aug-8d9y1q" class="trc_popover_aug_container">
  <div id="tbl-aug-2e8dxn" class="trc_popover_aug_container">
    <div id="tbl-aug-h7cua9" class="trc_popover_aug_container
       ">
      <div class="trc_popover trc_popover_fade trc_bottom">
        <div class="trc_popover_arrow"></div>
        <iframe frameborder="0" scrolling="no" src="
           javascript:void(0)" style="width: 100%" title="">
        </iframe>
      </div>
    </div>
  </div>
</div>
```

Listing 3.16: The same <*iframe*> from the ClimaTempo site after implementing a *title* attribute

The juxtaposition of Listings 3.15 and 3.16 underscores the vital role that title attributes play in augmenting the accessibility of iframes. Initially, the absence of a title (Listing 3.15) leaves a gap in accessibility, making it difficult for users of assistive technologies to grasp the iframe's function or content. The subsequent addition of a title attribute (Listing 3.16), as demonstrated with the ClimaTempo website, exemplifies a significant stride towards inclusivity. By furnishing a permanent, understandable description of the iframe's content, this simple yet impactful enhancement directly addresses the limitations previously encountered and aligns with WCAG 2.2 guidelines for creating web content that is accessible to all users, particularly those utilizing screen readers. These examples from ClimaTempo not only highlight the efficacy of incorporating title attributes into iframes but also serve as a compelling call to action for web developers to prioritize accessibility in their designs, ensuring that every element on a webpage contributes to a more navigable and comprehensible digital environment for users of all abilities.

4

Implementing the Catalog

To automate our catalog, we now introduce a plugin for the Visual Studio Code IDE called *a11y-refactoring*. The numeronym “a11y” is a shortened way of referring to the word “accessibility”. It is formed by the letter “a”, followed by the number “11” and the letter “y”. This number is often used in web development and technology to refer to digital accessibility. It’s a quick and concise way to convey the idea of making digital products, services, and content accessible to everyone, regardless of their abilities or disabilities.

4.1 a11y-refactoring plugin

The main objective of our plugin is to help developers to fix accessibility issues in their web projects by performing automatic analysis and transformations on the HTML source code. It implements our preliminary catalog presented previously, following three main steps:

1. **Code Analysis**: The plugin scans the project’s HTML code, identifying patterns that match the left-hand side of our transformation rules. This analysis is thorough and tailored to detect a wide range of accessibility issues;

2. **Storage and Identification**: All identified patterns are used as a reference point for the plugin, ameliorating any elements that require attention and refactoring from an accessibility point of view;

3. **Automatic Transformation**: For each identified item, we apply the corresponding transformation, replacing or modifying the code according to the right-hand side of our rules. This step is essential to ensure that the changes we implement effectively improve the site’s accessibility.

Developed in JavaScript, our plugin operates by directly manipulating the HTML Document Object Model (DOM). This approach allows for real-time analysis and transformation of the web project's HTML structure. By working directly with the DOM, the plugin can efficiently apply the necessary modifications to enhance accessibility, ensuring that the updates are both immediate and accurate. This method is particularly effective for dynamic content and interactive web applications, where accessibility improvements need to be integrated seamlessly with existing functionalities.

After the analysis, the plugin builds on our preliminary catalog to automate transformations in the HTML code, making content more accessible for people using screen readers. In the plugin, we are especially considering the experience of users with visual impairments or blindness, to ensure better interaction with assistive technologies.

Additionally, the plugin requests additional information from developers, such as the language for the *lang* attribute. For the title attribute of the `<a>` tag, we introduce standard text informing about the opening of a new window or tab.

Our plugin is open source, and we invite the community to contribute and improve it. Detailed information about how to install the plugin, how it works, and the transformations it performs is available on our website.[1]

4.2 Architecture of the Accessibility Refactoring Extension

The architecture of the *a11y-refactoring* extension within the Visual Studio Code (VSCode) development environment is illustrated in Figure 4.1. This structure is devised to aid accessibility refactoring on websites, with a particular focus on blind users. The diagram delineates the interconnected components and the stages of the refactoring process, as described below:

In **Context 1**, we encounter the VSCode Extension's user interface, where the extension is initially loaded and awaits a refactoring command from the user. Upon receipt of such a command, the extension activates the refactoring process.

Within **Context 2**, the File Access Controller takes center stage, first checking whether there is an active editor with an open HTML file. Upon verification, the active document is fetched and prepped for refactoring, establishing the necessary parameters. Subsequently, the main processor is invoked to commence the refactoring operations.

In **Context 3**, the Main Processor, along with the Utility Functions, carries out the refactoring process. The Utility Functions, each responsible for a specific accessibility task such as adding *alt* attributes to images or *lang* to HTML tags, are called in sequence. Following the execution of these functions, the critical section of the process is reached, where the File Writer saves the modified HTML content back to the file system.

This architectural diagram provides a clear view of the *a11y-refactoring* extension's com-

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

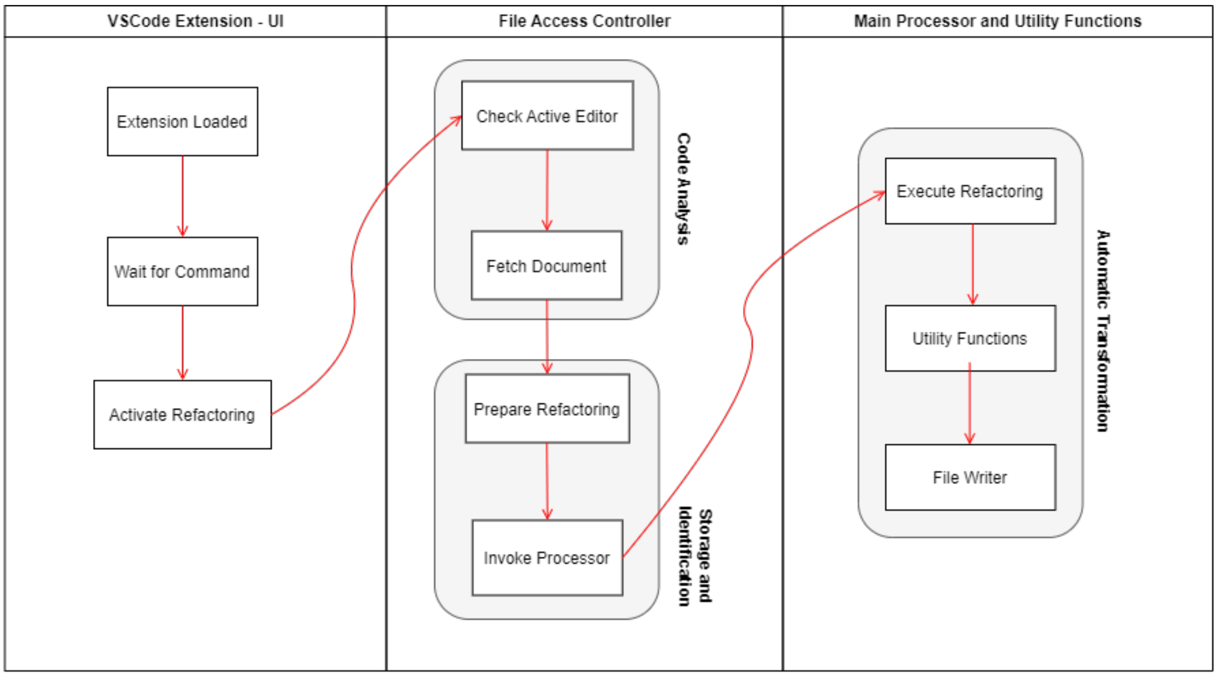


Figure 4.1: *a11y-refactoring* tool execution flow

ponents and their interplay, showcasing a methodological path to inclusive development and highlighting the extension as an essential tool in any web developer's toolkit who is mindful of the importance of digital accessibility.

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.[1] 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 Chapter 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*).

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

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*. These simple websites were specifically created for the second part of the study (**RQ2**) to control and isolate the effects of our improvements.

We structure our study using the Latin Square Design, as described by Box et al. (2005). This way, we have two treatments: with and without our improvements. In this design, we dispose the blind participants 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 5.1 presents our design. Notice that Person 2 first executes the study on the website 'Bakery' with our improvements, and then on the website 'Pet Shop' without our improvements. Person 1 executes in the opposite way: first without (in the website 'Bakery') and then with the improvements (in the website 'Pet Shop').

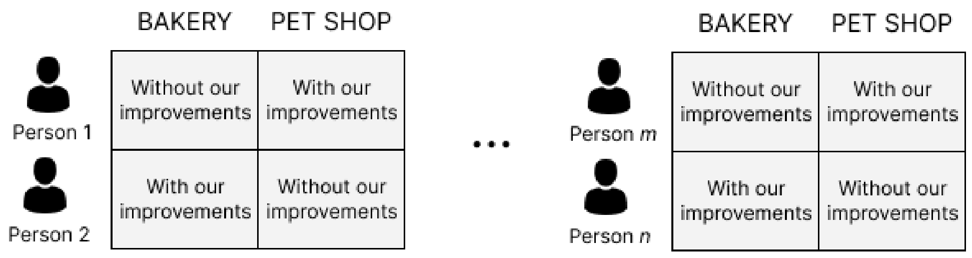


Figure 5.1: Latin Square design used in our study

This design is interesting to avoid learning effects. For example, if we let a person 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).

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.[2]

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

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 5.2, Figure 5.3, and Figure 5.4 help to answer **RQ1**. Notice that the number of critical errors has decreased in all three evaluators after submitting the websites with our improvements.

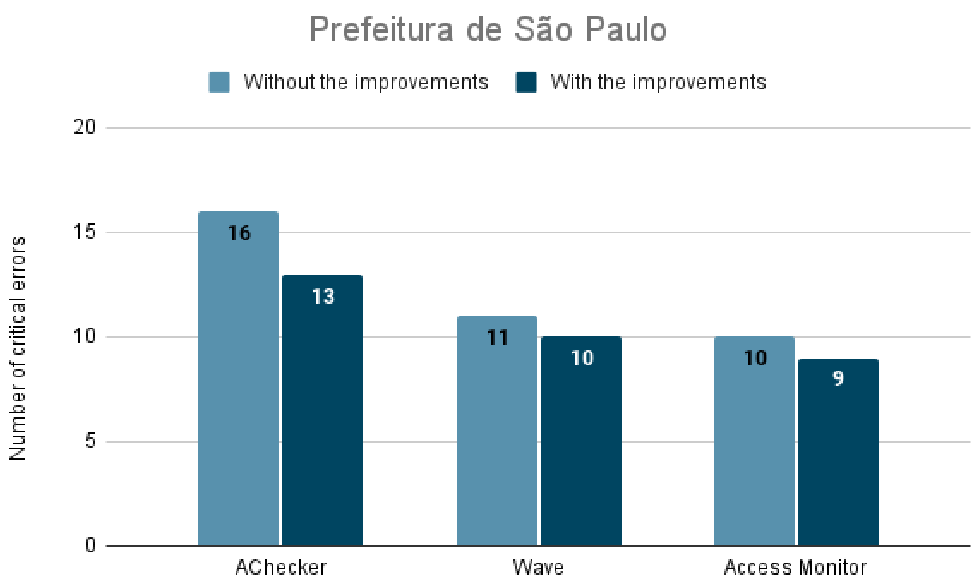


Figure 5.2: Number of critical errors with and without improvements on the Prefeitura de São Paulo website

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.

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.*"

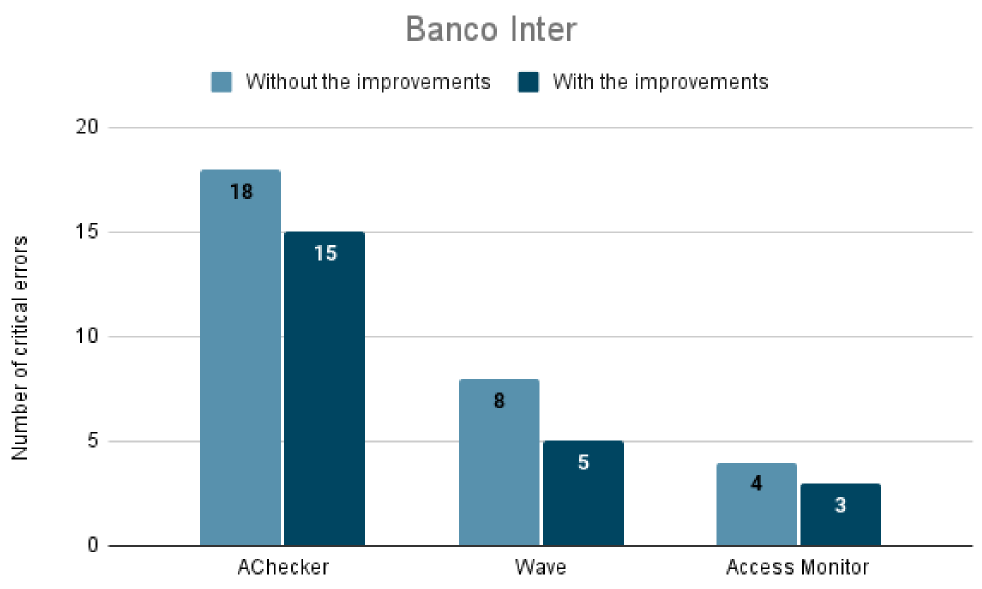


Figure 5.3: Number of critical errors with and without improvements on the Banco Inter website

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 specifically for **RQ2**. 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 who 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.

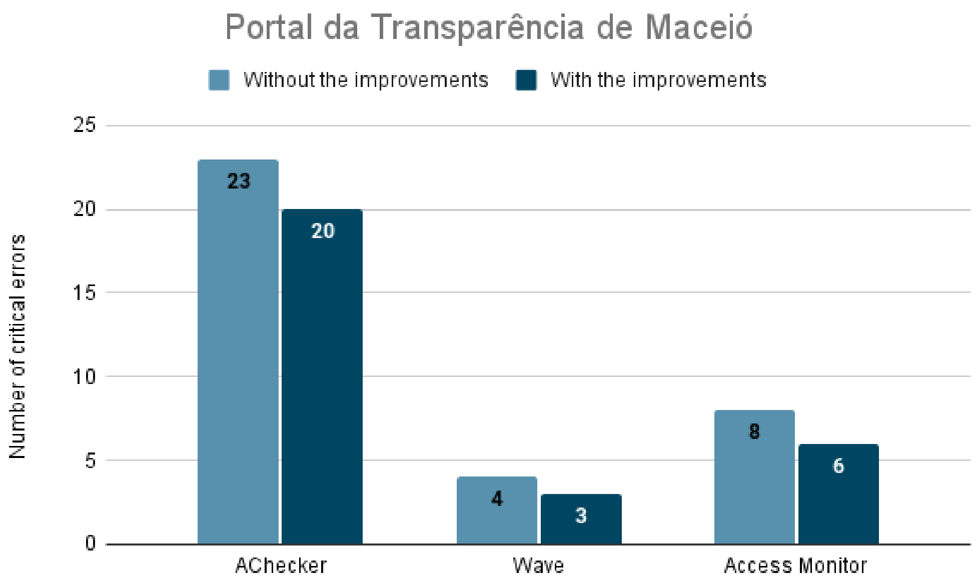


Figure 5.4: Number of critical errors with and without improvements on the Portal da Transparência de Maceió website

6

Related Work

Evaluating website accessibility often entails checking compliance with established standards, such as WCAG 2.2. This assessment can be conducted either on the server side, within a web development environment, or on the client side, in a browser environment, employing a range of methodologies from automatic checks to semi-automatic and manual approaches.

Prior works have focused on enhancing navigability for users with low vision using screen magnification Syed Masum Billah (2018). However, while interaction with interfaces receives considerable attention in usability studies, these approaches predominantly address visual accessibility on the web without encompassing the comprehensive operability of a page's interface. Our work diverges by concentrating on HTML elements to augment navigability, akin to the issues encountered by screen reader users. Additionally, researchers have delved into accessibility issues impacting the sequential order of user interface elements during navigation Alshayban et al. (2020), albeit their focus was restricted to Android devices. Our work does not focus on mobile devices.

Previous research has underscored the significance of image descriptions and their quality's impact on blind people's comprehension of web content Rodríguez Vázquez (2016); Jeong et al. (2023); Calvo et al. (2016), highlighting the necessity of precise, informative alternative text to elevate web accessibility. This aligns with our findings, where the importance of the *alt* attribute was emphasized by blind users. Although generating alternative texts for images is not our primary focus, we aim to enhance web accessibility for blind users by incorporating not only the *alt* attribute but also other attributes and semantic structures.

A study illuminated accessibility challenges for screen reader users, probing the difficulties in interpreting web page semantics through assistive technologies Almasoud and Mathkour (2019). Their findings stressed the need for properly structured HTML and the use of ARIA attributes for more effective screen reader interactions. However, this study developed an extension for the Google Chrome browser, introducing browser-level limitations. In contrast, our tool, a plugin for the VSCode IDE, boasts a broader scope and circumvents the requirement for API queries through a Google Chrome extension.

Automatic tools, such as AChecker Web Accessibility,[1] play a pivotal role in allowing developers, designers, and webmasters to perform swift and efficient website accessibility assessments. These tools generate reports identifying accessibility issues and suggesting fixes in line with WCAG 2.2 compliance levels A, AA, or AAA. However, these tools are unable to directly modify the server's source code, limiting their potential to enact immediate changes. In contrast, our tool distinguishes itself by facilitating direct alterations to the HTML code through the IDE, thereby streamlining the process of enhancing web accessibility.

Semi-automatic tools offer an interactive assessment of website accessibility. For instance, Vischeck[2] enables developers to simulate color perception for those with color blindness, facilitating manual color adjustment. However, it is limited to color adjustments and does not extend to semantic HTML element correction as our tool does.

The challenge remains with tools that cannot specify whether certain client-side elements of an HTML page violate WCAG success criteria. Tools focused on aspects like link context[3] can identify hyperlink issues but fail to diagnose problems in HTML nodes, such as the absence of essential attributes. Our tool addresses these deficiencies by correcting issues across HTML nodes, including alternative texts, website language, semantic tables, and more.

These studies underscore the ongoing need to explore and develop solutions that address both visual accessibility and the overall operability of interfaces. By integrating specific strategies for screen reader users and enhancing the semantic structure of web pages, we strive for a truly inclusive web that caters to all users, fostering a more accessible and navigable online experience.

[1] https://achecks.org/achecker/

[2] https://www.vischeck.com/vischeck/

[3] https://chromewebstore.google.com/detail/link-checker/olcpkmmoifipcklgnphbhdhbpfniijmb?hl=pt-br

7

Conclusion

In this work, we present a comprehensive approach to enhance the accessibility of websites, going beyond traditional practices. Our strategy involves a preliminary catalog of transformations for HTML code, aiming to make websites more accessible by focusing on elements like links, pictures, iframes, and tables. This catalog has been integrated into a plugin for the Visual Studio Code IDE, which is a part of our innovative method that transcends merely assessing WCAG compliance for HTML client pages. We aim to scrutinize and rectify the HTML source code directly.

The foundation of our approach is the development of *a11y-refactoring*, an automated tool designed to refine web accessibility in the code. This tool significantly reduces errors on websites by applying pattern matching techniques to the left-hand side and right-hand side of the source code. However, it is essential to note that not every technique based on success criteria can be fully implemented with this tool due to limitations in software detection capabilities for certain aspects of these techniques.

To evaluate the effectiveness of our catalog and tool, we conducted a double empirical study. Firstly, we assessed the extent to which our improvements enhanced website accessibility using online evaluators. Secondly, we surveyed blind individuals to determine if they noticed the improvements. The results were promising: all raters reported a decrease in critical errors post-implementation of our plugin, and seven out of eight blind participants could identify specific improvements, such as more structured tables or explicit site language definitions.

Our approach marks a significant advancement in web accessibility, especially in the often-neglected area of source code optimization. It emphasizes the importance of continued investment in web accessibility improvements, particularly through automated tools. Looking ahead, we intend to refine our catalog further, extend its applicability to server-side (e.g., PHP, Django) and client-side frameworks (e.g., Angular, React), and conduct more empirical studies to solidify its effectiveness.

References

Americans with disabilities act. https://www.ada.gov/. Accessed February 10, 2024.

Accessibility for ontarians with disabilities act. https://www.aoda.ca/. Accessed February 10, 2024.

European accessibility act. https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A32019L0882. Accessed February 10, 2024.

United nations convention on the rights of persons with disabilities. https://www.un.org/development/desa/disabilities/convention-on-the-rights-of-Persons-with-disabilities.html. Accessed February 10, 2024.

Suliman K. Almasoud and Hassan I. Mathkour. Instant adaptation enrichment technique to improve web accessibility for blind users. In *Proceedings of the 2019 3rd International Conference on Information System and Data Mining*, ICISDM 2019, page 159–164, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450366359. **DOI** 10.1145/3325917.3325931. URL https://doi.org/10.1145/3325917.3325931.

Abdulaziz Alshayban, Iftekhar Ahmed, and Sam Malek. Accessibility issues in android apps: State of affairs, sentiments, and ways forward. In *Proceedings of the ICSE*, page 1323–1334. ACM, 2020. ISBN 9781450371216. **DOI** 10.1145/3377811.3380392. URL https://doi.org/10.1145/3377811.3380392.

George E. P. Box, J. Stuart Hunter, and William G. Hunter. More than one blocking component: Latin squares. In *Statistics for Experimenters: Design, Innovation, and Discovery*, chapter 4.4, pages 157–162. Wiley-Interscience, 2005.

Brazil. Law no. 10,098, of december 19, 2000, 2000. URL http://www.planalto.gov.br/ccivil_03/LEIS/L10098.htm. Establishes general norms and basic criteria for promoting accessibility for people with disabilities or reduced mobility, and other provisions. Diário Oficial da República Federativa do Brasil. Accessed May 16, 2023.

Brazil. Law no. 13,146, of july 6, 2015, 2015. URL http://www.planalto.gov.br/ccivil_03/_ato2015-2018/2015/lei/l13146.htm. Establishes the Brazilian Law for the Inclusion of Persons with Disabilities (Statute of Persons with Disabilities). Diário Oficial da República Federativa do Brasil. Accessed May 16, 2023.

Rocio Calvo, Faezeh Seyedarabi, and Andreas Savva. Beyond web content accessibility guidelines: Expert accessibility reviews. In *Proceedings of the DSAI*, page 77–84. ACM, 2016. ISBN 9781450347488. **DOI** 10.1145/3019943.3019955. URL https://doi.org/10.1145/3019943.3019955.

World Wide Web Consortium. Web accessibility booklet - w3c, November 2013. URL https://www.w3c.br/pub/Materiais/PublicacoesW3C/cartilha-w3cbr-acessibilidade-web-fasciculo-I.html".

World Wide Web Consortium. Web content accessibility guidelines (wcag) 2.2. Technical report, October 2023a. URL https://www.w3.org/TR/WCAG22.

World Wide Web Consortium. Web content accessibility guidelines (wcag) 2.2. Technical report, October 2023b. URL https://www.w3.org/TR/WCAG22/#dfn-change-of-context.

Ana Ferreira and Márcio Ribeiro. Making websites more accessible for blind people with automatic html code transformations. *Congresso Brasileiro de Software: Teoria e Prática (CBSoft)*, pages 70–79, 2023. **DOI** https://doi.org/10.5753/cbsoft$_{e}stendido$.2023.235743.

Hyeonhak Jeong, Minki Chun, Hyunmin Lee, Seung Young Oh, and Hyunggu Jung. Wataa: Web alternative text authoring assistant for improving web content accessibility. In *Companion Proceedings of IUI*, page 41–45. ACM, 2023. ISBN 9798400701078. **DOI** 10.1145/3581754.3584127. URL https://doi.org/10.1145/3581754.3584127.

Movimento Web Para Todos. The number of brazilian websites approved in all accessibility tests has fallen compared to last year and is even less than 1%, June 2022. URL https://mwpt.com.br/numero-de-sites-brasileiros-aprovados-emtodos-os-testes-de-acessibilidade-tem-que
Accessed May 10, 2023.

Silvia Rodríguez Vázquez. Measuring the impact of automated evaluation tools on alternative text quality: A web translation study. In *Proceedings of the W4A*. ACM, 2016. ISBN 9781450341387. **DOI** 10.1145/2899475.2899484. URL https://doi.org/10.1145/2899475.2899484.

Donald E. Porter IV Ramakrishnan Syed Masum Billah, Vikas Ashok. Steeringwheel: A locality-preserving magnification interface for low vision web browsing. *ACM DL*, pages 3–9, 2018. **DOI** https://doi.org/10.1145/3173574.3173594.

Deque Systems. axe accessibility linter, 2021. [Accessed Aug 02, 2023].

Max van der Schee. Web accessibility, 2018. [Accessed Aug 02, 2023].