Usage of Artificial Intelligence Tools to Improve Digital Web Accessibility

Todor Todorov[0000-0002-2443-4618]

Institute of Mathematics and Informatics, Bulgarian Academy of Sciences, Sofia, Bulgaria
St. Cyril and St. Methodius University of Veliko Tarnovo, Veliko Tarnovo, Bulgaria
t.todorov@ts.uni-vt.bg

Abstract. Accessibility of websites considers challenges related to creation of web pages adapted to user with different types of impairments. The goal of web accessibility is to remove barriers that may prevent people with disabilities from accessing or interacting with web content effectively. The paper presents some web accessibility standards and challenges related to their complying. A special attention is paid to some popular artificial intelligence tools and how they could be used to solve accessibility issues on web sites development.

Keywords: Website Accessibility, Accessibility Evaluation, Artificial Intelligence, Web technologies.

1 Introduction

Accessible websites should have functionality adapted to user with different types of disabilities. It plays a crucial role in ensuring that individuals with disabilities can perceive, understand, navigate, and interact with online content effectively, thereby promoting inclusion and equal participation in the digital society (Web Accessibility Best Practices, n.d.), (Accessibility, n.d.). The establishment of accessibility guidelines, such as the Web Content Accessibility Guidelines (WCAG), by the World Wide Web Consortium (W3C) has been instrumental in providing a framework for developers and designers to create accessible web content (Web Content Accessibility Guidelines (WCAG), n.d.). Despite significant progress in this area, challenges remain in achieving universal accessibility, including the need for greater awareness, improved tools and resources, and ongoing research to address evolving technology and user needs.

The evolution of accessibility guidelines has been shaped by a growing recognition of the importance of web accessibility and advancements in technology. Early initiatives, such as the Web Accessibility Initiative (WAI) launched by W3C in 1997, laid the groundwork for subsequent efforts to establish comprehensive standards for web accessibility (Web Accessibility Initiative (WAI), n.d.). The development of WCAG, initially released in 1999 and subsequently updated in 2008 and 2018, marked a significant milestone in providing a set of principles, guidelines, and success criteria for creating accessible web content.

Digital Presentation and Preservation of Cultural and Scientific Heritage. Conference Proceedings. Vol. 14, Sofia, Bulgaria: Institute of Mathematics and Informatics – BAS, 2024. ISSN: 1314-4006, eISSN: 2535-0366

Research in web accessibility explores various aspects, including:

- **Evaluation Methods:** Developing and evaluating automated and manual techniques for assessing website accessibility compliance with WCAG (Ara et al., 2024; Nuñez et al., 2019)
- **User Experience (UX):** Investigating the impact of accessibility features on user experience for both users with and without disabilities (Sauer et al., 2020) Aizpurua et al., 2016).
- **Emerging Technologies:** Exploring accessibility considerations for new web technologies like Single Page Applications (SPAs) (Findel & Navon, 2015; Gavrilă et al., 2019).
- **Assistive Technologies:** Researching the effectiveness and limitations of various assistive technologies used by people with disabilities (Yesilada & Harper, 2019).

In Section 2 is made an overview of the methods and principles related to building of accessible applications. Sections 3 presents HTML, CSS, and JavaScript examples of web accessibility challenges. In Section 4 are considered some popular AI tools and how they could be used to solve accessibility issues in web applications. Concrete concepts and results are presented.

2 Methodologies for Accessibility

Web accessibility research revolves around the Web Content Accessibility Guidelines (WCAG) established by the World Wide Web Consortium (W3C). WCAG outlines four core principles (WCAG Accessibility Principles, n.d.):

- **Perceivable**: Information and user interface (UI) components must be presented in a way that users can perceive.
- **Operable**: UI components and navigation must be operable using various input methods, including keyboards.
- **Understandable**: Information and the operation of the UI must be understandable.
- **Robust**: Content must be robust enough to be compatible with a wide range of assistive technologies.

Accessibility evaluation methodologies encompass a range of approaches, including manual inspection, assistive technology testing, and automated evaluation tools (Abascal et al., 2019). Manual inspection involves human evaluators examining web content to identify accessibility barriers and issues. While this method provides valuable insights into the user experience, it can be time-consuming and subjective. Assistive technology testing involves using specialized software and devices, such as screen readers and keyboard navigators, to assess the accessibility of web content from the perspective of users with disabilities. Automated evaluation tools leverage algorithms and heuristics to scan web pages for accessibility violations automatically (Sanchez-Gordon & Luján-Mora, 2017; Abu Doush et al., 2023). While these tools offer efficiency and scalability, they may not capture all accessibility issues accurately and may generate false positives or false negatives.

Accessible design practices have been shown to have a positive impact on user experience outcomes, including usability, satisfaction, and task performance (Pickering, 2016). Empirical studies have demonstrated that incorporating accessible design features, such as descriptive alternative text for images, logical document structure, and keyboard navigation support, can improve the usability of websites for users with disabilities and enhance the overall user experience for all users. Accessible design principles, such as clarity, consistency, and flexibility, contribute to making web content more understandable, navigable, and adaptable to diverse user needs and preferences.

Despite the importance of web accessibility, numerous challenges and barriers persist in achieving universal accessibility. Technical challenges include the complexity of web technologies, compatibility issues with assistive technologies, and the dynamic nature of web content. Organizational challenges relate to the lack of awareness, resources, and incentives for prioritizing accessibility in web development processes. Attitudinal barriers stem from misconceptions and stereotypes about disabilities, which may lead to a lack of commitment to accessibility or resistance to implementing accessibility measures. Overcoming these challenges requires a multi-faceted approach that addresses technical, organizational, and attitudinal factors and fosters collaboration among stakeholders to promote a culture of accessibility.

Emerging technologies, such as artificial intelligence (AI) and machine learning, hold promise for advancing website accessibility and addressing existing challenges. AI-driven tools and techniques can automate accessibility testing, content remediation, and personalized user interactions, thereby enhancing the efficiency and effectiveness of accessibility efforts (Nacheva & Jansone, 2023). For example, AI-powered algorithms can analyse web content for accessibility issues, suggest remediation strategies, and adapt content presentation based on user preferences and interaction patterns. Future research directions may focus on developing AI-driven solutions that integrate seamlessly into existing web development workflows, address emerging accessibility challenges, and promote inclusive design practices across diverse contexts and platforms.

3 Challenges and Barriers to Accessibility Related to Web Technologies

In addition to broader challenges, specific issues within the realms of HTML, CSS, and JavaScript present unique accessibility barriers. Next are provided examples of specific challenges and strategies addressing HTML, CSS, and JavaScript accessibility problems. HTML, as the foundation of web content, may lack semantic structure or fail to include essential elements, such as proper headings, labels for form fields, or alternative text for images, which are critical for users relying on screen readers or other assistive technologies. For example, consider the following HTML code snippet:

```
<div class="button" onclick="submitForm()">Submit</div>
```

In this example, a ** ** element styled as a button is used for form submission without providing any accessible name or role. Screen reader users may not be able to identify the purpose of the button, leading to confusion and usability issues.

Common accessibility problems associated with specific code types also include:
Missing alternative text for images: Screen readers cannot interpret images without alt text, creating barriers for visually impaired users. Following code provides no description of the image for screen reader users:

```
<img src="image.jpg">
```

Semantic HTML misuse: Using non-semantic elements like ** ** for navigation menus hinders screen reader navigation.

```
<div id="navigation">
   <a href="/">Home</a>
   <a href="/about">About Us</a>
</div>
```

This code uses a div for navigation. A more semantic approach would be to use a **<nav>** element and nest li elements for each link.

Similarly, CSS can introduce accessibility issues when used improperly, such as overriding default focus styles or relying heavily on visual cues alone to convey information, making it difficult for users with visual impairments to perceive content. For instance, consider the following CSS code snippet:

```
.hidden {
    display: none;
}
```

Here, the **.hidden** class is used to visually hide content from sighted users. However, screen reader users will still perceive this content unless additional measures, such as applying **aria-hidden="true"**, are taken to ensure it is not announced to assistive technologies.

Insufficient colour contrast: Text and background colour combinations with low contrast make content difficult to read for users with low vision:

```
p {
  color: #DDD;
  background-color: #EEE;
}
```

This code displays light grey text on a light grey background, creating low contrast and making the text difficult to read.

JavaScript, while enhancing interactivity and functionality, can also pose challenges if not implemented accessibly, such as keyboard traps, lack of focus management, or

dynamic content updates that are not announced to screen reader users. For example, consider the following JavaScript code snippet:

```
function openModal() {
    document.getElementById('modal').style.display = 'block';
}
```

This code snippet opens a modal dialog when invoked. However, it does not manage keyboard focus properly, potentially trapping keyboard users within the modal without a means to exit, violating accessibility guidelines. As web technologies continue to evolve, addressing HTML, CSS, and JavaScript accessibility problems remains a critical focus area for researchers and practitioners. Emerging trends such as Web Components and Single Page Applications (SPAs) introduce new challenges and opportunities for accessibility. Web Components offer a modular approach to web development but require careful consideration of accessibility implications, such as ensuring custom elements expose accessible roles, states, and properties to assistive technologies. For example, consider the following Web Component implementation with accessibility considerations:

```
<custom-button label="Submit" onclick="submitForm()"></custom-button>

class CustomButton extends HTMLElement {
    connectedCallback() {
    this.setAttribute('role', 'button');
    this.setAttribute('tabindex', '0');
    this.addEventListener('click', this.onclick);
    }
```

In this example, the Web Component encapsulates button behaviour while ensuring proper semantics with the usage of attributes **role** and **tabindex**. This approach promotes accessibility and encapsulation, enhancing maintainability and reusability. Similarly, SPAs, while enhancing user experience, can present accessibility challenges related to dynamic content updates and navigation.

4 Accessibility and Artificial Intelligence

Contemporary AI models like ChatGPT 3.5 (ChatGPT, n.d.), Copilot (Copilot, n.d.) and Gemini (Gemini, n.d.) could be used to successfully generate source of simple and more complicated web applications. It is interesting to investigate to what extent this generated code observes the web accessibility principles.

One of the main accessibility principles is to add alternative text for all images in a web site. Two of the considered AI engines are used to generate such a text for two images prompting them with: “Generate alt text for the following image”. On Figure 1

and Figure 2 are presented the two images and the results. First image is an artwork from the Europeana database, and the second one is a popular image from the Internet.

Fig. 1. Test image 1.

Fig. 2. Test image 2.

In Table 1 are presented result from Gemini and Copilot engines related to generated alternative text. It could be concluded that both engines generate a relatively good versions of description text of the pictures content. Only notable difference is that Copilot recognized the exact number of cups on the Image 2.

Table 1. Generated alternative text.

	Gemini	Copilot
Image1	Colourful tulips and other flowers in a botanical drawing.	An arrangement of colourful flowers and plants in a vase
Image2	Multiple coffee cups on a wooden table	Seven empty coffee mugs on a wooden table.

Next test is to use AI tools to improve the accessibility of an HTML menu. On Figure 3 is shown the HTML code of the menu. It uses table tags which are not well interpreted by screen readers and that is why it is recommended not to use such an approach of navigation. The prompt to both ChatGPT and Gemini was to "Improve the accessibility of the following code":

```
<table>
    <tr>
      <td><a href="Home.html">Home</a></td>
    </tr>
        <tr>
      <td><a href="about.html">About</a></td>
    </tr>
    <tr>
      <td><a href="contacts.html">Contacts</a></td>
    </tr>
</table>
```

Fig. 3. Navigation HTML code.

On Figure 4 and Figure 5 are presented results from the optimized for accessibility code generated by Gemini and ChatGPT respectively. Gemini only recognizes that the table is used as a navigation element and adds a suitable caption to it. It does not replace it with another set of tags that are more suitable for screen readers. On the other hand, ChatGPT improves accessibility by using semantic HTML elements for navigation instead of a **<table>**. It uses **<nav>** element along with an unordered list (**<ul>**) and list items (**<li>**).

```
<table>
  <caption>Main Navigation</caption>
  <tbody>
    <tr>
      <th><a href="Home.html">Home</a></th>
    </tr>
    <tr>
      <th><a href="about.html">About</a></th>
    </tr>
    <tr>
      <th><a href="contacts.html">Contacts</a></th>
    </tr>
  </tbody>
</table>
```

Fig. 4. Navigation HTML code after Gemini optimization.

```
<nav>
  <ul>
    <li><a href="home.html">Home</a></li>
    <li><a href="about.html">About</a></li>
    <li><a href="contacts.html">Contacts</a></li>
  </ul>
</nav>
```

Fig. 5. Navigation HTML code after ChaGPT optimization.

```
function openModal() {
      document.getElementById('modal').style.display = 'block';
  }
```

Fig. 6. JavaScript dialog function source code.

Another test for web accessibility features of AI engines is performed using the JavaScript code from Figure 6.

To improve accessibility, it should be added appropriate keyboard and screen reader support to modal functionality. Additionally, it's important to be ensured that the modal can be navigated and dismissed using the keyboard. On Figure 7 is presented the optimized code the generated by ChatGPT.

```
function openModal() {
    let modal = document.getElementById('modal');
    modal.style.display = 'block';
    modal.setAttribute('aria-modal', 'true');
    modal.setAttribute('aria-hidden', 'false');
    modal.focus();
    modal.addEventListener('click', closeModal);
    document.addEventListener('keydown', closeModalOnEscape);
}

function closeModal(event) {
        if (event.target === modal) {
                modal.style.display = 'none';
                modal.setAttribute('aria-hidden', 'true');
                modal.removeEventListener('click', closeModal);
                document.removeEventListener('keydown', closeModalOnEscape);
        }
}

function closeModalOnEscape(event) {
        if (event.key === 'Escape') {
                closeModal(event);
        }
}
```

Fig. 7. JavaScript dialog function source code after ChaGPT optimization.

The code generated by ChatGPT successfully uses ARIA attributes to improve the accessibility of the modal dialog. It also adds handlers to keyboard events and the click of the **ESCAPE** key is processed as an option for the dialog closing.

5 Conclusion

Website accessibility is a multifaceted and evolving field that requires ongoing research, collaboration, and innovation to ensure equal access to digital content for all users. Continued efforts to embrace emerging technologies and promote accessibility best practices are essential for realizing the vision of a web that is accessible, usable, and enjoyable for everyone. In the paper is made an overview of methodologies and

technologies for building accessible web applications. Special attention is paid on the AI tools and their usage for accessibility improvements. Examples with source code analysis using different web technologies (HTML, CSS, JavaScript) are presented.

The conclusions from the research show that ChatGPT 3.5 produces best accessibility improvements compared to other tools involved in the experiments.

It will be interesting as a future research to be investigated the performance of more advanced AI tools like ChatGPT 4.0, Gemini Advanced and Copilot Pro.

Acknowledgements.

This research was funded by the National Science Fund of Bulgaria (scientific project "Digital Accessibility for People with Special Needs: Methodology, Conceptual Models and Innovative EcoSystems"), Grant Number KP-06-N42/4, 08.12.2020.

References

Abascal, J., Arrue, M., & Valencia, X. (2019). Tools for Web Accessibility Evaluation. In Y. Yesilada, & S. Harper (Eds.), *Web Accessibility. Human–Computer Interaction Series* (pp 479–503). Springer. https://doi.org/10.1007/978-1-4471-7440-0_26

Abu Doush, I., Sultan, K., Al-Betar, M., Almeraj, Z., Alyasseri, Z., & Awadallah, M. (2023). Web accessibility automatic evaluation tools: to what extent can they be automated? *CCF Transactions on Pervasive Computing and Interaction, 5*(9), 288–320. https://doi.org/10.1007/s42486-023-00127-8

Accessibility. (n.d.). https://www.interaction-design.org/literature/topics/accessibility

Aizpurua, A., Harper, S., & Vigo, M. (2016). Exploring the relationship between web accessibility and user experience. *International Journal of Human-Computer Studies*, *91*, 13-23. https://doi.org/10.1016/j.ijhcs.2016.03.008

Ara, J., Sik-Lanyi, C., & Kelemen, A. (2024). Accessibility engineering in web evaluation process: a systematic literature review. *Universal Access in the Information Society*, *23*, 653–686. https://doi.org/10.1007/s10209-023-00967-2

ChatGPT. (n.d.). https://chat.openai.com/

Copilot. (n.d.). https://copilot.microsoft.com/

Findel, H., & Navon, J. (2015). A Test Environment for Web Single Page Applications (SPA). In *Proceedings of the 11th International Conference on Web Information Systems and Technologies - WEBIST* (pp. 47-54). https://doi.org/10.5220/0005428000470054

Gavrilă, V., Bajenaru, L., & Dobre, C. (2019). Modern Single Page Application Architecture: A Case Study. *Studies in Informatics and Control, 28*(2), 231-238. https://doi.org/10.24846/v28i2y201911

Gemini. (n.d.). https://gemini.google.com/app

Nacheva, R., & Jansone, A. (2023). Heuristic Evaluation of AI-Powered Web Accessibility Assistants. *Baltic Journal of Modern Computing, 11*(4), 542-557. https://www.bjmc.lu.lv/fileadmin/user_upload/lu_portal/projekti/bjmc/Contents/11_4_02_Nacheva.pdf

Nuñez, A., Moquillaza, A., & Paz, F. (2019). Web Accessibility Evaluation Methods: A Systematic Review. In A. Marcus, & W. Wang, (Eds.) *Design, User Experience, and Usability. Practice and Case Studies. HCII 2019. Lecture Notes in Computer Science, vol. 11586* (pp. 226–237). Springer, Cham. https://doi.org/10.1007/978-3-030-23535-2_17

Pickering, H. (2016). *Inclusive design patterns : coding accessibility into web design.* Smashing Magazine.

Sanchez-Gordon, S., & Luján-Mora, S. (2017). A Method for Accessibility Testing of Web Applications in Agile Environments. In *Proceedings of the 7th World Congress for Software Quality (WCSQ)*. En proceso de publicación.(citado en la página 13, 15, 85). http://sandrasanchez.blog.epn.edu.ec/wp-content/uploads/sites/141/2017/04/WCSQ-2017-Sanchez-Gordon-Lujan-Mora-Accessibility-Testing.pdf

Sauer, J., Sonderegger, A., & Schmutz, S. (2020). Usability, user experience and accessibility: towards an integrative model. *Ergonomics*, *63*(10), 1207–1220. https://doi.org/10.1080/00140139.2020.1774080.

WCAG Accessibility Principles. (n.d.). https://www.w3.org/TR/UNDERSTANDING-WCAG20/intro.html

Web Accessibility Best Practices. (n.d.). https://www.freecodecamp.org/news/web-accessibility-best-practices/

Web Accessibility Initiative (WAI). (n.d.). https://www.w3.org/WAI/

Web Content Accessibility Guidelines (WCAG). (n.d.). https://www.w3.org/TR/WCAG21/

Yesilada, Y., & Harper, S. (2019). *Web Accessibility - A Foundation for Research.* Springer. https://doi.org/10.1007/978-1-4471-7440-0

Received: March 15, 2024
Reviewed: April 09, 2024
Finally Accepted: April 30, 2024