Exhibit A

Karl Groves, *Sole reliance on accessiBe will not be sufficient in ensuring full and equal access to a website* (Nov. 1, 2020)

Sole reliance on accessiBe will not be sufficient in ensuring full and equal access to a website

Karl Groves

November 1, 2020

Introduction

accessiBe is a product marketed to customers with claims that it can automatically bring a website into compliance with the Americans with Disabilities Act (ADA) and the Web Content Accessibility Guidelines (WCAG). The product is provided to customers as a snippet of JavaScript code which must be added to each page of the customer's site. That JavaScript snippet loads an application, hosted on accessiBe servers. This application provides a series of features ranging from controls that allow the user to modify the website's appearance. It also provides a series of disability specific "profiles" which provide enhancements aimed at the challenges faced by those specific disability types. In some circumstances, the product also attempts to repair the site's underlying accessibility issues.

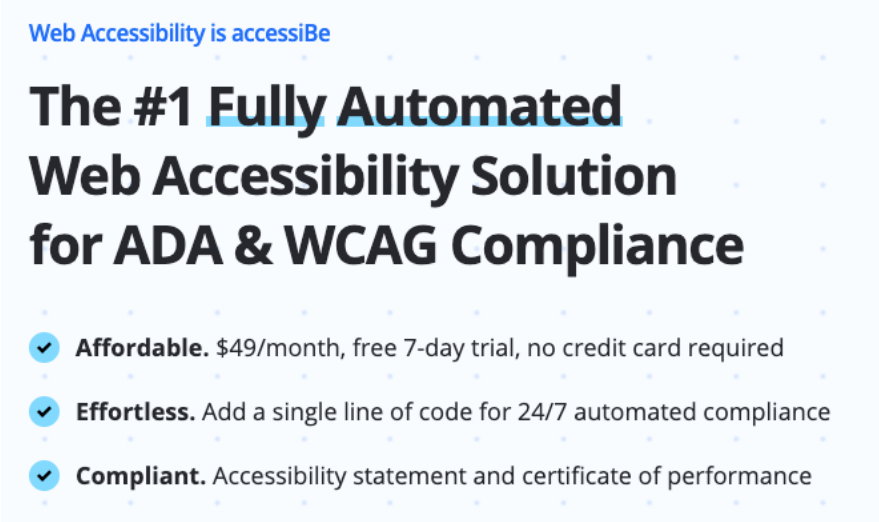


Figure 1: Screenshot from accessiBe website showing its claims regarding compliance

This document describes the ways in which the accessiBe product does not and/ or cannot ensure full and equal access to a website. In addition, it provides evidence that the accessiBe widget itself adds net-new accessibility problems to the customers' site.

Definition of accessibility

Accessibility can be viewed as the "ability to access" and benefit from some system or entity. The concept focuses on enabling access for people with disabilities, or special needs, or enabling access through the use of assistive technologies (https://en.wikipedia.org/wiki/Accessibility). The ADA has its own definition of disability allowing it to define those who are protected by the law. This definition makes clear that protected populations are not limited to those with only specific disabilities, but rather includes any "physical or mental impairment that substantially limits one or more major life activities" and are also not limited to disabilities that are only permanent. (https://www.ada.gov/pubs/adastatute08.htm#12102).

Requirements for ADA compliance

With respect to ADA compliance on the Web, the US Department of Justice says: "Although the language of the ADA does not explicitly mention the Internet, the Department has taken the position that title III covers access to Web sites of public accommodations. The Department has issued guidance on the ADA as applied to the Web sites of public entities, which includes the availability of standards for Web site accessibility". (https://www.ada.gov/regs2010/titleIII_2010/titleIII_2010_regulations.pdf)

All further information and advice available from the US DOJ cites the Web Content Accessibility Guidelines (WCAG) from the W3C's Web Accessibility Initiative as the standard for use in determining a system's accessibility. WCAG is further incorporated by reference into Section 508 of the Rehabilitation Act and is used as the technical standard for nearly 2-dozen accessibility laws throughout the world (https://www.w3.org/WAI/policies). As a consequence of the above, conformance with WCAG, at the AA Level, is the standard required in legal settlements and judgments for lawsuits in the US alleging noncompliance with ADA.

In essence, a website must conform to WCAG Level AA in order to comply with the ADA.

Background on overlays

"Overlays" are third-party products which are embedded into a website with the purpose of modifying the website's presentation layer in a way that repairs or overcomes the site's accessibility issues. Such products have existed on the market since 1999 with projects like ReadSpeaker. Additional products, like BrowseAloud and ESSENTIALAccessibility entered the market in the early 2000s. All such products claimed to transform the sites onto which they were embedded so that the site would be accessible after implementation. The increased legal activity in the United States in the last few years has also created market opportunities for others to release similar products. Some of those products, like Make-Sense/ a11yAble, AudioEye, and accessiBe provide a broader array of features than the early products like BrowseAloud. While early products stated that their text-to-speech capabilities created a fully accessible site, the more recent products include the ability to also transform the UI in ways that are claimed to bring the site into compliance.

Conformance cannot be automated

To understand why all overlay products, including accessiBe, fail in their goal of automatically creating a compliant website, it is important to consider the complexities inherent to accessibility.

The WCAG Standard, last updated on June 5, 2018, presents 13 broadly worded guidelines for creating accessible content. Those guidelines are categorized into one of 4 key principles. The guidelines are further broken down into 73 Success Criteria.

> "For each guideline, testable success criteria are provided to allow WCAG 2.0 to be used where requirements and conformance testing are necessary such as in design specification, purchasing, regulation, and contractual agreements."
> (https://www.w3.org/TR/WCAG21/)

The normative portion of the WCAG standard is 105 pages, when printed. Additional informative documents: "How to Meet WCAG", "Understanding WCAG", and "Techniques and Failures" are, cumulatively, over 2000 pages when printed. The fact that the informative content that accompanies the standard is so long is demonstrative of the fact that conformance with WCAG is not so straightforward that it can be automated.

In its description of conformance, the WCAG documentation states:

> "Conformance to a standard means that you meet or satisfy the 'requirements' of the standard. In WCAG 2.0 the 'requirements' are the Success Criteria. To conform to WCAG 2.0, you need to satisfy the Success Criteria, that is, there is no content which violates the Success Criteria."
> (https://www.w3.org/TR/UNDERSTANDING-WCAG20/conformance)

To claim conformance to a specific Level of WCAG, you must satisfy all the criteria within that level:

- For Level A conformance (the minimum level of conformance), the Web page satisfies all the Level A Success Criteria, or a conforming alternate version is provided.
- For Level AA conformance, the Web page satisfies all the Level A and Level AA Success Criteria, or a Level AA conforming alternate version is provided

(https://www.w3.org/TR/WCAG21/#conformance)

Furthermore: Conformance (and conformance level) is for full Web page(s) only and cannot be achieved if part of a Web page is excluded.

In other words, if WCAG Level AA is the standard used to determine ADA compliance, then it is to be understood that **all** Level AA Success criteria must be met on each page of the site. The converse is also

true: **not meeting one or more Level AA Success criteria in a page means that the page is not compliant.**

The breadth of the problem

As part of the research for this report, I selected 50 websites of accessiBe customers and ran automated testing against an average of 20 pages on each site. At the time of testing, each of these websites were verified to have the accessiBe widget on the site. The purpose of this testing was to gather basic data on the scope and nature of issues which exist on a typical site. The full list of sites is listed in Appendix A.

Total Issues Discovered Per Site	1. MIN: 399 2. MAX: 12,077 3. MEAN : 2754 4. MEDIAN: 2334
Issues Per Page	5. MIN: 28 6. MAX: 932 7. MEAN : 155 8. MEDIAN: 124 9. MODE: 127
Average Page Density (Defined as number of issues per kilobyte of source)	10. MIN: 11% 11. MAX: 176% 12. MEAN : 56% 13. MEDIAN: 51% 14. MODE: 49%
Average issues per Page: Images and other non-text content	15. MIN: .26 16. MAX: 118 17. MEAN : 14 18. MEDIAN: 6 19. MODE: N/A
Average issues per page: Forms	20. MIN: 1 21. MAX: 77 22. MEAN : 12 23. MEDIAN: 5 24. MODE:2
Average issues per page: Keyboard Accessibility and Focus Control	25. MIN: 1 26. MAX: 123 27. MEAN : 19 28. MEDIAN: 15 29. MODE: 21

Percentage of pages with zero issues	30. 0%

Figure 2: High level automated testing results

As the table above shows, the sites tested contained a significant number of issues with 0% of pages being completely error-free. The data above does not show any significant divergence from what has been found across the broader set of websites I have tested. In other words, the accessiBe customer sites are neither better nor worse than the broader Web as a whole.

The data above makes clear that accessibility on the Web is a serious challenge.

Fundamental truths on machine testing, automatic transformations, and accessibility

To automatically bring a non-compliant site into compliance with WCAG, it stands to reason that such a product must first be able to automatically detect every individual instance of non-compliance. For instance, to repair an issue with a Keyboard Trap (WCAG 2.1.2) such a product must be capable of detecting the existence of a Keyboard Trap. From there, the tool must be able to determine whether the Keyboard Trap is essential based on the current context or whether it is a trap with no other means of escaping. Finally, once such determination has been made, the tool must also decide on where to send keyboard focus next based on the user's interaction.

That example is only one of many complex and currently insurmountable challenges facing testing tools and automation. More examples:

1. **For 1.1.1 Non-text Content:** It is possible to detect a missing text alternative for an image, but it is not possible to know, with 100% certainty whether a text alternative, when supplied, is an accurate alternative for the meaning intended by the document author.
2. **For 1.2.2. Captions (Prerecorded):** It is possible to know, for certain audio & video embedding methods (i.e., native HTML5), whether a programmatically associated captions file exists, but it is not possible to know this for other mechanisms (Flash). It is also not possible to know if the captions don't already exist as "open captions" on the video itself. Furthermore, it is impossible to automatically test for the quality of those captions.

In fact, it is impossible to accurately judge full conformance for any of WCAG's 73 success criteria using machine testing alone. Therefore, it stands to reason that automated repair of nonconformance is impossible as well. Quite simply, some WCAG SCs are too complex to accurately test or repair (3.3.3, 1.4.1) while others are too subjective (1.1.1).

In addition to the complexity and subjectivity of testing for WCAG, using an external tool for testing & repairing of content other than HTML, CSS, and JS is impossible as well. This is due to the architecture of the Web and of Web pages. While a third-party JavaScript application such as accessiBe can gain access to all objects present in the DOM, some on-page resources cannot be manipulated with JavaScript. **Content presented in Flash, Java, Silverlight, or PDF cannot be assessed or modified by third-party JavaScript.** This is a fact that accessiBe readily admits on their own Terms of Service:

> "By way of example, accessiBe Systems do not support other components, such as Canvas, Flash and/or SVG." They also state "...accessiBe does not remediate PDF files or create subtitles for videos…" their Terms also further say "The Company does not undertake that the Licensee Website will be 100% accessible at any given moment, owing to factors such as Licensee changes made to the Licensee Website, issues originating in the Licensee Website and /or limitations stemming from technological reasons."

accessiBe further disclaim their marketing department's statements of automated conformance by stating:

> "The functionality of the accessiBe Systems requires that the Licensee Website in which they are embedded be websites based solely on HTML files and tags, and that the source code be written according to the Standard of the World Wide Web Consortium ("W3C"), without any errors or validation warning in W3C's troubleshooting inspections; please note that Licensee changes to such website may impact the functionality of the Service."
> https://accessibe.com/terms-of-service

The impracticality of the above requirement in accessiBe's Terms of Service is demonstrated in the statistics available from the W3C service they cite in the above passage. The statistics, available at https://validator.w3.org/nu/stats.html indicate that approximately 80% of all tested documents contain validation errors that would, in turn, invalidate a customer's claims that the accessiBe wasn't doing its job. However, markup validity in and of itself is unlikely to be an accessibility issue, evidenced by the fact that WCAG removed such a requirement when going from version 1.0 to 2.0.

By requiring that customer websites' markup consist of valid HTML, and by failing to support PDF, video, or other "limitations stemming from technological reasons" accessiBe is indicating that it recognizes the fact that it is impossible to fully automate compliance.

Key examples of accessiBe's failure to provide a completely accessible experience

This section describes an array of examples, for a variety of different content types, where accessiBe does not effectively repair accessibility issues.

AccessiBe is unable to correct accessibility issues in images and other non-text content

Images are an integral part of the web and are commonplace on most websites. The image element (`<img>`) has an `alt` attribute which allows an author to provide a clear, concise alternate description of the image's content. Other methods exist to provide text-based alternatives including, where necessary, mechanisms which will cause assistive technologies to ignore images for which no text-based alternative is necessary.

Alternate descriptions allow low and no vision users to understand the image's content and, importantly, intent behind its placement. Without meaningful alternate descriptions, assistive technology users will not be able to understand website content the same way non-assistive technology users can.

Alternate descriptions should communicate information critical to understanding what the image is meant to convey. An alternate text description should also capture pertinent thematic details, such as the color, texture, and materials for product photography. Doing so ensures that the experience has a parity in quality with what a non-assistive technology user experiences. A full description of the decision making needed to create an effective text alternative is available at: https://www.w3.org/WAI/tutorials/images/decision-tree/ and a fully detailed description is available in the HTML5 specification, Section 4.8.4.4 Requirements for providing text to act as an alternative for images (https://html.spec.whatwg.org/multipage/images.html#alt)

accessiBe purportedly uses "computer vision" to automatically create and apply alternate descriptions (https://accessibe.com/product/artificial-intelligence). This automation of alternate descriptions may fail in one (or more) of the following instances:

1. accessiBe may not detect the presence of an image, and therefore will not assign it an image description.
2. accessiBe creates an image description, but the image description does not accurately represent the image's content and meaning to the end user.
3. The existing image description is insufficient and accessiBe does not overwrite it.

In each case, these issues are a failure of Web Content Accessibility Guidelines 2.1 Success Criterion 1.1.1 Non-text Content (https://www.w3.org/TR/WCAG21/#non-text-content). Following are a representative sampling of instances of these issues.

Situations in which accessiBe does not detect the presence of an image

In certain cases, accessiBe is unable to detect the presence of an image and, as a result, does not attempt to provide a text alternative for it.

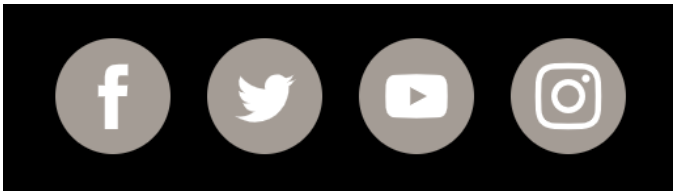


Figure 3: Example from Belkin https://www.belkin.com/us/

Code for one of the icons above:

```
<svg viewBox="0 0 512 512"><path d="M211.9 197.4h-
36.7v59.9h36.7V433.1h70.5V256.5h49.2l5.2-59.1h-54.4c0 0 0-22.1 0-33.7
0-13.9 2.8-19.5 16.3-19.5 10.9 0 38.2 0 38.2 0V82.9c0 0-40.2 0-48.8 0
-52.5 0-76.1 23.1-76.1 67.3C211.9 188.8 211.9 197.4 211.9
197.4z"></path></svg>
```

- Original text alternative: None provided
- Alternate description with accessiBe's Blind User Profile active: None provided
- Issues:
    - The images are defined using SVG, not an image element. SVGs are an alternate way to display images, typically used for icons and simple illustrations. SVGs also require an alternate description, provided using techniques that differ from the image element's alt attribute.
    - SVG without an alternate description cannot be understood by low and no vision users navigating via assistive technology.
    - Each of these SVG images does not have an alternate description.

- accessiBe does not detect these SVG images, and therefore does not provide an alternate description. accessiBe explicitly states that it does not support SVG (https://accessibe.com/terms-of-service) and, as a result, cannot bring into compliance any site which makes use of SVG.

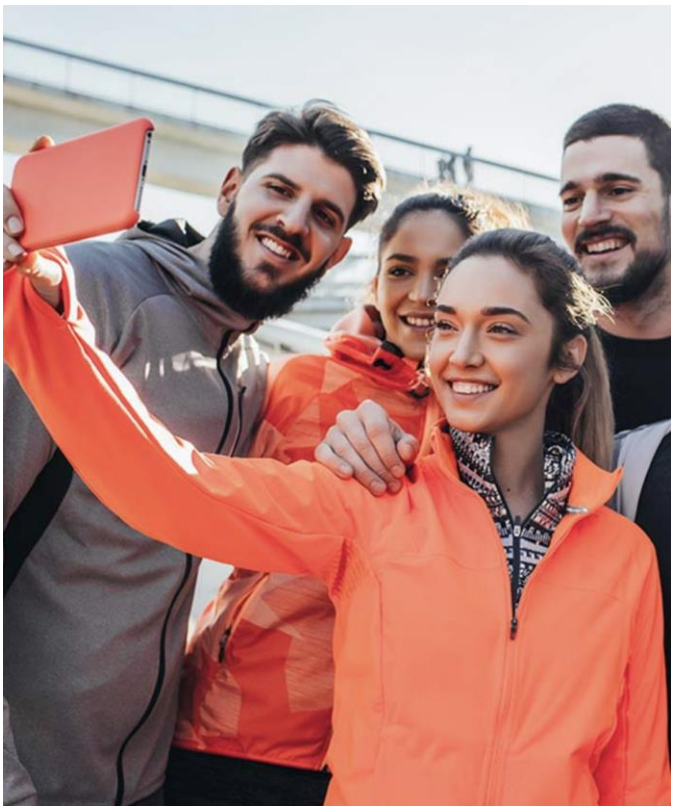


Figure 4: Example from Belkin Student Discount https://www.belkin.com/us/studentdiscount/

```
<iframe
src="https://connect.studentbeans.com/v2/belkin/us?stb_offer_path=http
s%3A%2F%2Fwww.belkin.com%2Fus%2Fstudentdiscount%2F&amp;validate_iframe
=true" width="100%" height="720" frameborder="0"
seamless="seamless"></iframe>
```

- Original text alternative: None provided
- Alternate description with accessiBe's Blind User Profile active: None provided
- Issues:
  - This image is part of an embedded widget provided by StudentBeans, a third-party service.

- This image does not have an alternate description provided by StudentBeans.
- accessiBe does not detect images inside of embedded widgets, and therefore does not provide an alternate description.

Situations in which accessiBe creates an inaccurate image description

Bella Lace Wedding Dress

US $198.00

Figure 5: Example from Kiyonna: Featured Styles > Office Chic

https://www.kiyonna.com/plus-size-clothing/plus-size-work-clothes.html

- Original text alternative: None provided
- Text alternative with accessiBe's Blind User Profile active: "Grass nature and summer".
- Issues: accessiBe's machine-based description does not sufficiently communicate image content and purpose.

Flirty Flounce Wrap Dress - Sale!!

US $60.00

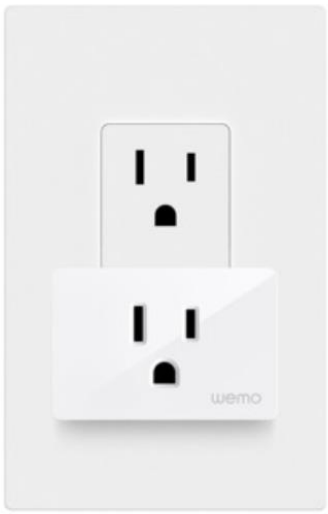
Figure 6: Example from Kiyonna: Featured Styles > Office Chic

https://www.kiyonna.com/plus-size-clothing/plus-size-work-clothes.html

- Original text alternative: None provided
- Text alternative with accessiBe's Blind User Profile active: "Fashion woman and girl"
- Issues: accessiBe's machine-based description does not sufficiently communicate image content and purpose.

Figure 7: Example from carousel on https://www.kiyonna.com/behind-the-seams/protective-face-masks/

- Original text alternative: “” (in other words, it was purposefully left empty)
- Text alternative with accessiBe’s Blind User Profile active: “BTS-2”
- Issues:
    - Alt description has been intentionally left blank but is not decorative.
    - The nulled alt description does not sufficiently describe the image’s content.
    - accessiBe description does not sufficiently communicate image content and purpose.
        - In this instance, it appears accessiBe’s automation logic cannot identify significant shapes within the image, and defaults to using a modified version of the uploaded filename, “BTS-2-500x280.jpg”. If the filename does not describe the image’s contents, the description will be insufficient.



Wemo WiFi Smart Plug

20% Off $19.99 ~~$24.99~~

Figure 8: Example from https://www.belkin.com/us/exclusive-deals/

- Original text alternative: None provided
- Text alternative with accessiBe's Blind User Profile active: "Processing the data, please give it a few seconds..."
- Issues: accessiBe description has failed to load and is using the alt description to communicate a status message. This does not sufficiently communicate the image's content and purpose.

CAR CHARGERS

Figure 9: Example from https://www.belkin.com/us/products/

- Original text alternative: None provided
- Text alternative with accessiBe's Blind User Profile active: "belkin. Power electricity and addiction"
- Issues: accessiBe's machine-based description does not sufficiently communicate image content and purpose

Figure 10: Example from https://annieselke.com/c/pineconehill

- Original text alternative: "" (in other words, it was purposefully left empty)
- Text alternative with accessiBe's Blind User Profile active: "Fas blogfooter1"
- Issues: accessiBe description does not sufficiently communicate image content and purpose. In this instance, it appears accessiBe's automation logic cannot identify the text used in the image, and defaults to using a modified version of the uploaded filename, "`FAS_BlogFooter1?fmt=png-alpha`". Since the filename does not describe the image's contents, the description is insufficient.

Situations in which accessiBe does not correct an inaccurate image description

Figure 11: Example from https://www.kiyonna.com/

- Original text alternative: "Treat Yourself Sale"

- Text alternative with accessiBe's Blind Users (Screen-reader) mode active: "Treat Yourself Sale"
- Issues: the accessiBe product did not correct the insufficient alt attribute value supplied by the customer site
    - Does not provide information about the $30 separates.
    - Does not provide information about the $60 separates.
    - Does not provide information about the shop sale call to action.

Figure 12: Screenshot from Kiyonna site. All content in this screenshot is a single image.

- Original text alternative: "Masker-AID Masks"
- Text alternative with accessiBe's Blind Users (Screen-reader) mode active: "Masker-AID Masks"
- Issues: the accessiBe product did not correct the insufficient alt attribute value supplied
    - Does not provide information about how Masker-AID is by Kiyonna.
    - Does not provide information about the headline, "You (M)asked, we listened".
    - Does not provide information about the description, "Made in the USA with 3 layers of 100% Soft Cotton! They are breathable, reversible, reusable, and washable."
    - Does not provide information about the shop masks call to action.
    - Does not provide information about the exclusion message, "(This item is excluded from all promotions and codes)".
    - Does not provide information about the three labeled polaroids with product photos and models.

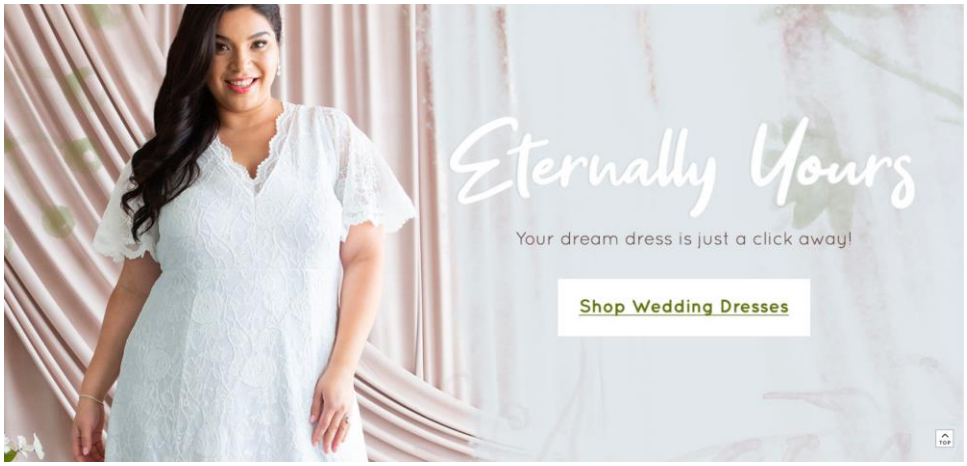


Figure 13: Screenshot from Kiyonna site. This entire screenshot is a single image

- Original text alternative: "New Wedding Dresses"
- Text alternative with accessiBe's Blind Users (Screen-reader) mode active: "New Wedding Dresses"
- Issues: the accessiBe product did not correct the insufficient alt attribute value supplied
    - Does not describe the subtitle, "Your dream dress is just a click away!"
    - Does not provide information about the presence of a model in a wedding dress.
    - Does not provide information about the Shop Wedding Dresses call to action.

BOOST↑CHARGE™ Wireless Charging Stand + Speaker

$49.99

Figure 14: Example from https://www.belkin.com/us/c/speakers-headphones/

- Original text alternative: “prodImage”
- Text alternative with accessiBe’s Blind User Profile active: “prodImage”
- Issues:
    - Existing alternate description does not sufficiently describe the image.
    - accessiBe does not update the insufficiently described image.

Figure 15: Example of logo on https://www.belkin.com/us/

- Original text alternative: “” (in other words, it was purposefully left empty)
- Text alternative with accessiBe’s Blind User Profile active: “”
- Issues:
    - Alt description has been left blank but is not decorative.
        - The nulled alt description does not sufficiently describe the image’s content.
    - A nulled title attribute is also present on this image element.
        - The nulled title attribute also prevents describing the image’s content.
    - accessiBe description does not sufficiently communicate image content and purpose, navigating home.

Figure 16: Example from https://www.belkin.com/us/p/P-BBM001/

- **Original text alternative:** “BOOST ↑ UP Wireless Charging Dock on bedside table”
- Text alternative with accessiBe’s **Blind User Profile** active: “BOOST ↑ UP Wireless Charging Dock on bedside table”
- Issues:
    - Existing alternate description incorrectly describes the image.
    - accessiBe does not update the incorrectly described image.

Figure 17: Example from https://annieselke.com/about-us

- Original text alternative: "Timeline 9"
- Text alternative with accessiBe's Blind User Profile active: "Timeline 9"
- Issues: the accessiBe product did not correct the insufficient alt attribute value supplied
    - Existing alternate description incorrectly describes the image.
        - The image contains both images of text and photos, and the presence of both are not announced.
    - accessiBe does not update the incorrectly described image.

The above demonstrates that accessiBe's approaches to handling text alternatives for images is ineffective and cannot correct inaccessibility thereof. Its attempts at creating text alternatives via image recognition fall short due to the inherent limitations of machine learning as well as the inappropriateness of such an approach in the first place.

An effective text alternative for non-text content is not merely a description of what an image contains but instead must consist of what *meaning* that the image contributes to the content. That meaning is heavily dependent on surrounding content and on the context in which it is used. Ultimately, only the web page's author truly knows what that is and using a machine to guess *meaning* is a mistake.

Even the world's largest software company cannot get this right, as evidenced by the screenshots below. Not long ago Microsoft Office added image recognition, which uses image recognition to attempt to make it easier for users to add text alternatives to images. As the screenshots below demonstrate, image recognition can often be wildly incorrect. In the first example, Microsoft's AI suggests "A close up of a speaker" for the image displaying the WiFi Smart Plug. In the second example which shows a car charger adapter from Belkin, Microsoft's AI merely says "Diagram"

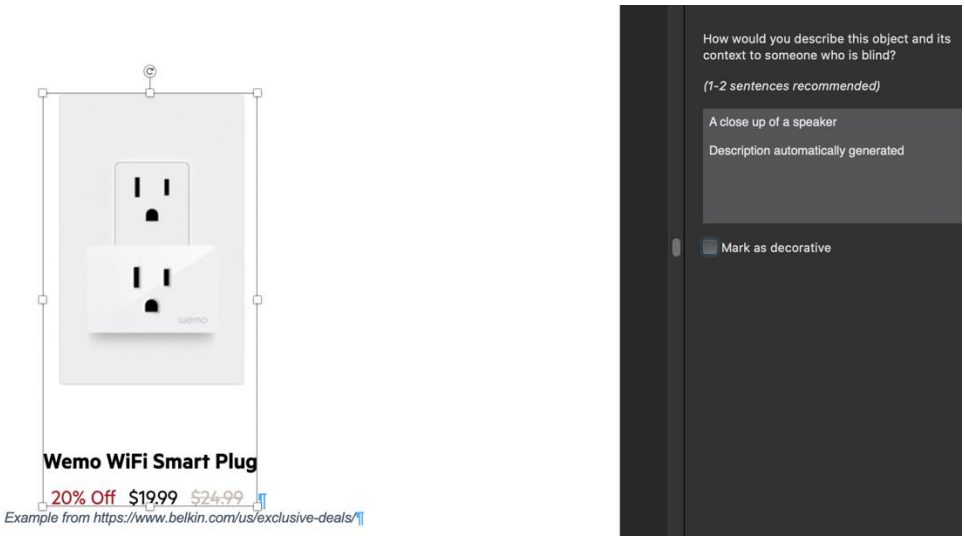


Figure 18: Screenshot of MS Word's alt text panel

Figure 19: Screenshot of MS Word's alt text panel

AccessiBe is unable to correct accessibility issues in Forms

Forms allow a user to act on website content and submit information for processing. They are a common fixture on websites, as they allow for things like eCommerce, job applications, email correspondence, and social media posting to be conducted.

Forms are constructed from a predefined set of HTML elements, including `<form>`, `<fieldset>`, `<label>`, `<input>`, `<textarea>`, `<select>` and `<button>`. These elements allow a user to conduct actions such as input text, select, and check options, upload files, and transmit information. These HTML elements also map to hooks provided by assistive technology such as screen readers. This allows low and no vision users to be able to understand the presence of a form, as well as what sorts of input it can receive.

accessiBe claims to use artificial intelligence to "solve" issues with "buttons," "forms," and "dropdowns"—all component parts found in forms. (https://accessibe.com/product/artificial-intelligence) The quality and accuracy of this automation is suspect given the results of before & after inspection of the sites of accessiBe customers.

A clear demonstration of accessiBe's inability to properly fix accessibility issues can be seen on the CandleScience website at the following URL: https://www.candlescience.com/quick_order/soap-fragrance-oil

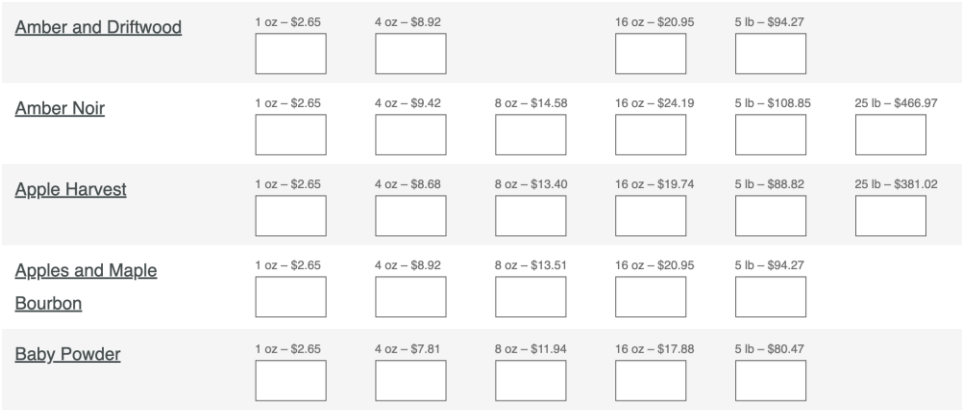


Figure 20: Screenshot of Bulk order grid on the CandleScience website

By default, there is no `<label>` element present for each input, nor is any other mechanism provided to create an accessible name for them. Without a sufficiently described accessible name for each input in the grid, it becomes difficult or impossible for low and no vision users to understand what each input is for.

accessiBe's approach appears to simply locate nearby text content and assign it to the closest applicable input to create an accessible name for the input via the `aria-label` attribute. For example, after enabling the Blind User profile on accessiBe, the inputs' markup are modified considerably as in the following example:

```
<input type="number" min="0" class="form-control" data-acsb-navigable="true" data-acsb-now-navigable="true" data-acsb-textual-type="null" data-acsb-validation-uuid="al3fvrgm6xsb" data-acsb-field-visible="true" aria-label="4 oz - $8.92" placeholder="4 oz - $8.92" data-acsb-tooltip="4 oz - $8.92">
```

The relevant accessibility "repair" is highlighted above. The accessiBe product has used that adjacent string of text (4 oz - $8.92) and applied it as the value for the `aria-label` attribute.

It also creates a placeholder attribute and a custom tooltip. The ultimate effect of this "repair" from accessiBe is:

1. Depending upon the screenreader user's settings, they may hear this same string of text - at minimum - twice, because the `aria-label` and `placeholder` values are the same.
2. The label provided is no better than the default behavior of all major screen readers, which means that this approach could be the same as doing nothing.

A label that consists solely of the size and price of the product is not an accurate or suitable label, as non-visual users will not know which product these features relate to. accessiBe does not associate each input with their corresponding fragrance listed to the left of the row(s) of inputs. With no association, a low or no vision user has no way of understanding what number of what fragrances they are purchasing.

There is an average of 126 instances of this type of issue on each of the 50 sites tested in Appendix A

Another example of accessiBe's inability to provide a good accessible name for a control can be seen in the main menu of the CandleScience website

Without the accessiBe widget enabled, the button element used to toggle open a search field does not have an accessible name. A `<button>` can be given an accessible name by providing text content in between the opening and closing <button> element tags. Other mechanisms exist to create an accessible name for buttons, such as by using `aria-label`. Buttons require an effective accessible name to create an announcement for screen readers about what functionality the button triggers. With no accessible name, a low or no vision user has no way of understanding what functionality the button triggers.

Enabling accessiBe's Blind User profile showed no modification to the button that would create an accessible name for this button.

There is an average of 28 instances of this type of issue on each of the 50 sites tested in Appendix A

AccessiBe is unable to correct accessibility issues in document heading structure

Semantic structure should be used to ensure that users can access all the content on a page. Regions, headings, lists, links, even paragraph tags can be used to simplify navigation through web content. Headings fulfill an important role for users who are blind, low-vision, and cognitively impaired. Visually, headings can convey to users what the page - and the content under each heading - is about. Screenreader users can also get this same understanding by listening to the headings. The level of each heading can be used to infer an "outline" of the page and its content. If headings are not identified or are identified out of order, blind users will be unable to understand the content structure in the same ways as a user who is not blind.

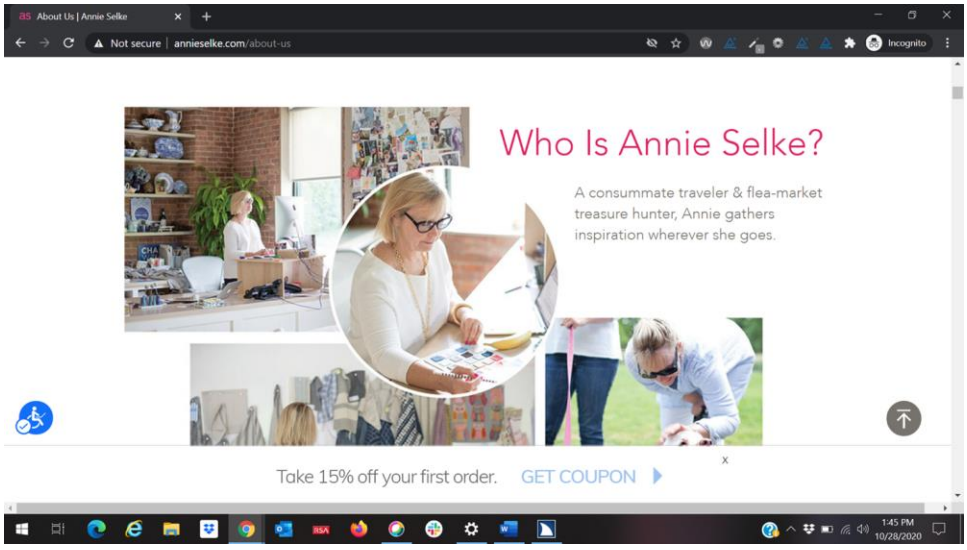


Figure 21: Screenshot of the AnnieSelke.com "About Us" page

When engaging with the "About Us" page on the Annie Selke page using JAWS alone, only “About Us” is identified as a heading. Users cannot navigate to other content on the page that is styled in a way that appears as a heading. With Accessibe's Blind User Profile enabled and JAWS turned on, “Who is Annie Selke” is identified with a role of heading, but it is assigned a Heading level 4. Users navigating through the heading structure alone are likely to find this content confusing.

However, even with this “enhancement,” not all the content that looks like a heading is identified as one:

1. “Mission & Philosophy” is not identified as a heading
2. “Commitment to Quality” is identified as two separate heading level 4s
3. “We Bring Happy Home” is not identified as a heading
4. “Who We Are” is not identified as a heading
5. Rather than sub-heading and content, the timeline is presented as a series of images identified as “Timeline 1,” “Timeline 2,” etc. through “Timeline 10.”

Like the example above, the product listing pages feature suboptimal heading structure. In this example, the page is reasonably structured in premise: It has a single H1, and the subheadings are used - despite skipping around. What’s noticeably challenging is that the page structure is not set up in a logical way. The left rail navigation comes after the main content in the body of the page, so the order is off. This is reflected in the outline view of the page:

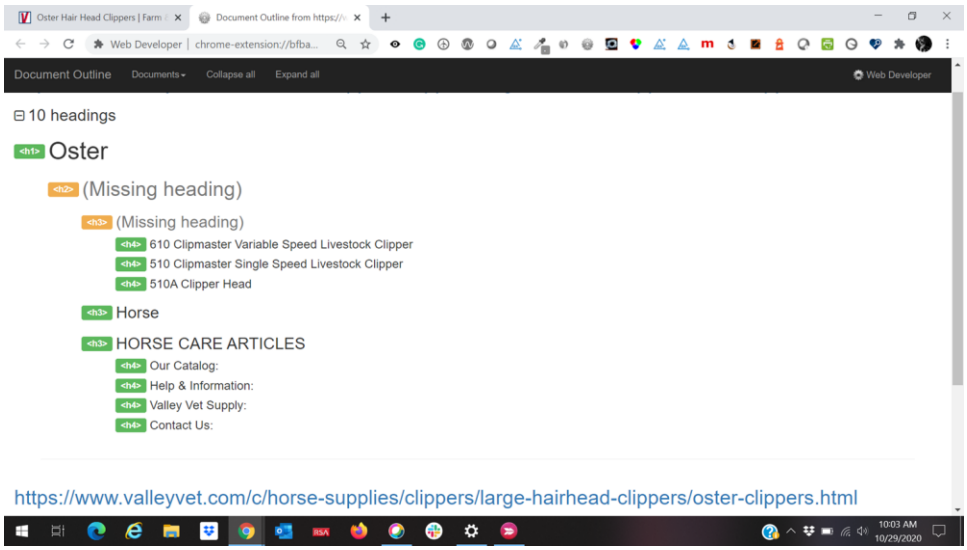


Figure 22: Screenshot demonstrating the poor heading structure on the ValleyVet site

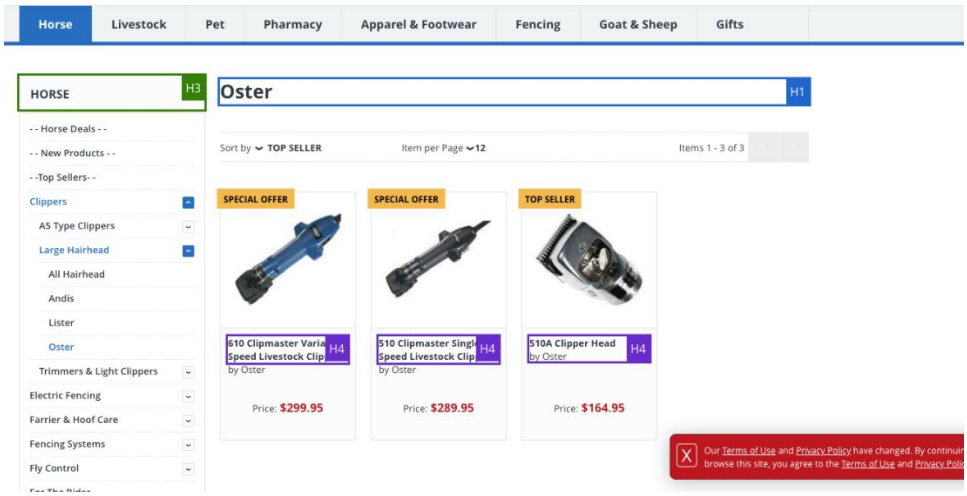


Figure 23: The same page as above, showing the headings in context

In both cases - be it the suboptimal heading structure and the poor content order, accessiBe does not fix these issues.

AccessiBe is unable to correct accessibility issues in Keyboard Navigation

On the ShowMeCables.com website, the "Shopping Options" sidebar on the left of the product list does not appear in the correct tab order (all shopping pages are affected). To navigate to this section, one must tab completely through the main content section. Only then is a keyboard user able to choose filters for sorting items. Some of the options, such as options for length and color, never achieve focus via keyboard at all.

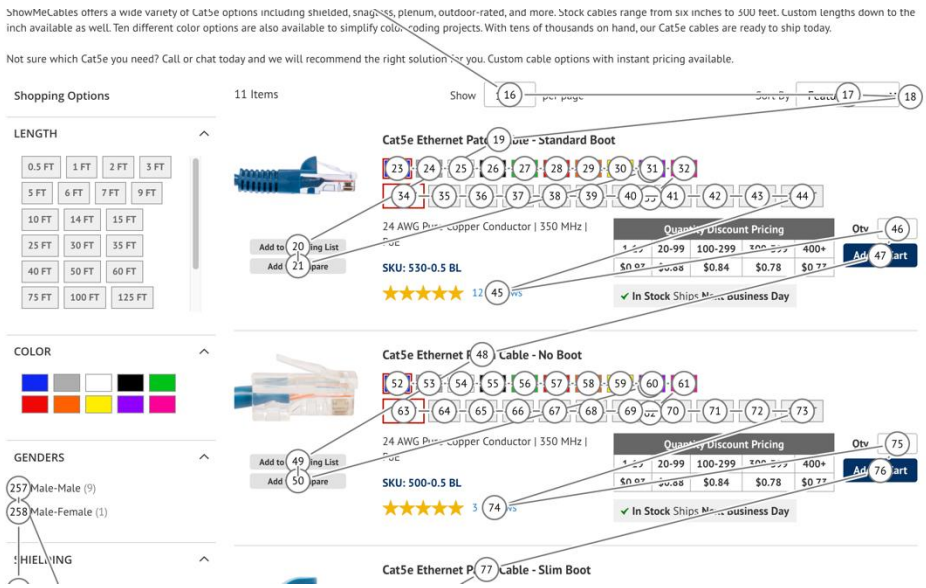


Figure 24: Screenshot from the ShowMeCables.com website showing the tab order.

As the screenshot above shows, the next tab stop after the global navigation and breadcrumb is #16: the control for users to select how many products to show per page. The filter controls on the left for "Genders" is #257.

Screen reader users and keyboard-only users expect that content on the Web will be in left-to-right and top-to-bottom order. On an interface like the one in the screenshot above, users expect to interact with the "Shopping Options" on the left before the list of products. Using the site with accessiBe's Blind User profile enabled this deficiency is not repaired.

On the same site, the "Custom Cable" building tool is wholly inaccessible to keyboard users. All operation in this tool requires the use of the mouse and this is not repaired by accessiBe's Blind User profile

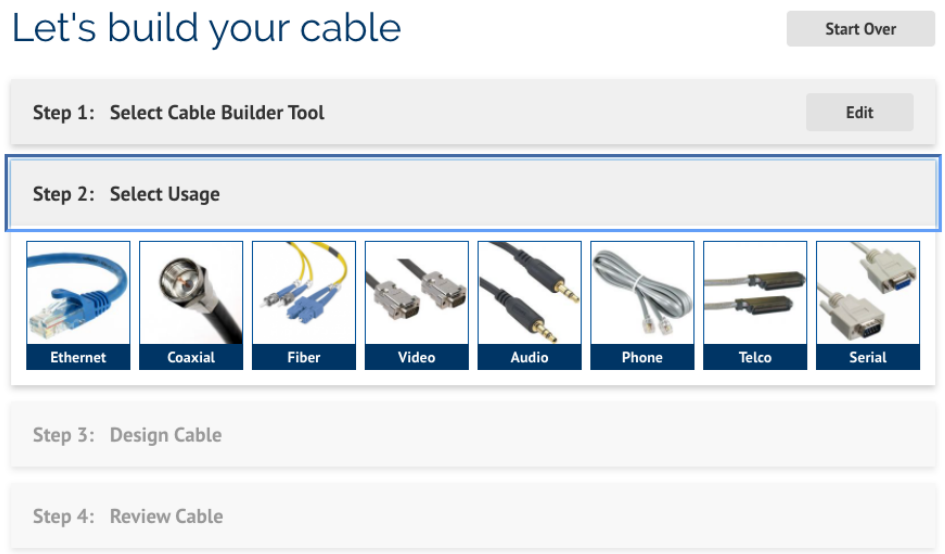


Figure 25: Screenshot of ShowMeCables.com's custom cable builder

Accessibility issues within the accessiBe widget

The previous several pages have demonstrated the vast array of accessibility problems that accessiBe does not correct. However, even if accessiBe was able to fix everything on the underlying page, the widget itself introduces its own set of new problems. The following is a brief, select list of issues within the accessiBe widget that constitute *new* accessibility issues that further bring the customer's site out of compliance.

Operating the Widget

To access the accessibility options and contents of the widget, the user must either click on the accessiBe icon or they can discover the content by tabbing into the page at which time an "Accessibility Feedback and Statement" control visually appears and is in the keyboard order. It is not clear from the accessible name of this control that it will lead to the various accessibility control options within the widget. Once it has been activated the user is taken directly to the statement which needs to be closed before the widget internals are exposed. This risks reducing the likelihood that end users will reach the widget to enable any of its features

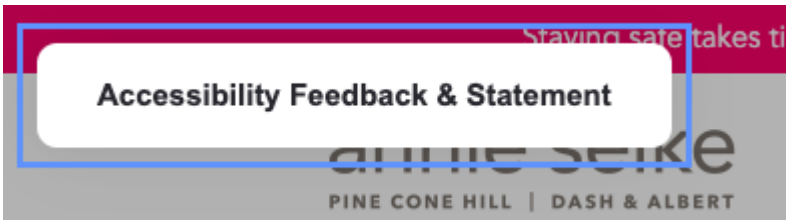


Figure 26: Screenshot of the initial tab stop in the product.

The control to select the language is a button with one default language selected (e.g., English) and no information is communicated about the ability to select from a list of languages. The button's label merely states the current language rather than convey that it allows users to switch the language.

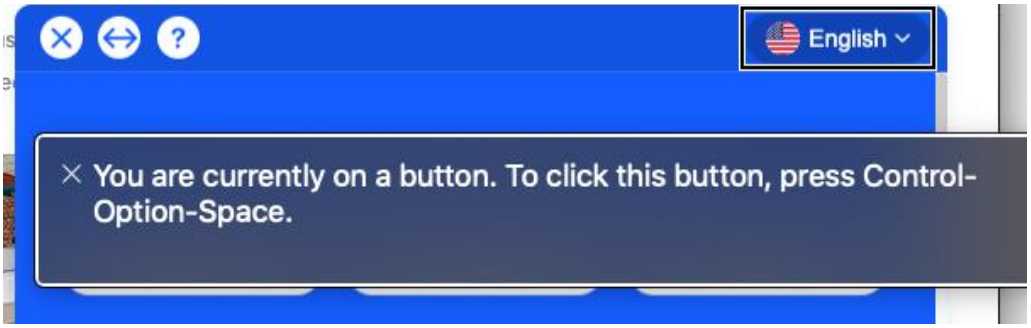


Figure 27: Screenshot of accessiBe widget and the caption output from Voiceover screenreader on MacOS

Once a user has activated the language button, a dialog opens that allows the user to select their desired language for the accessiBe widget's interface.

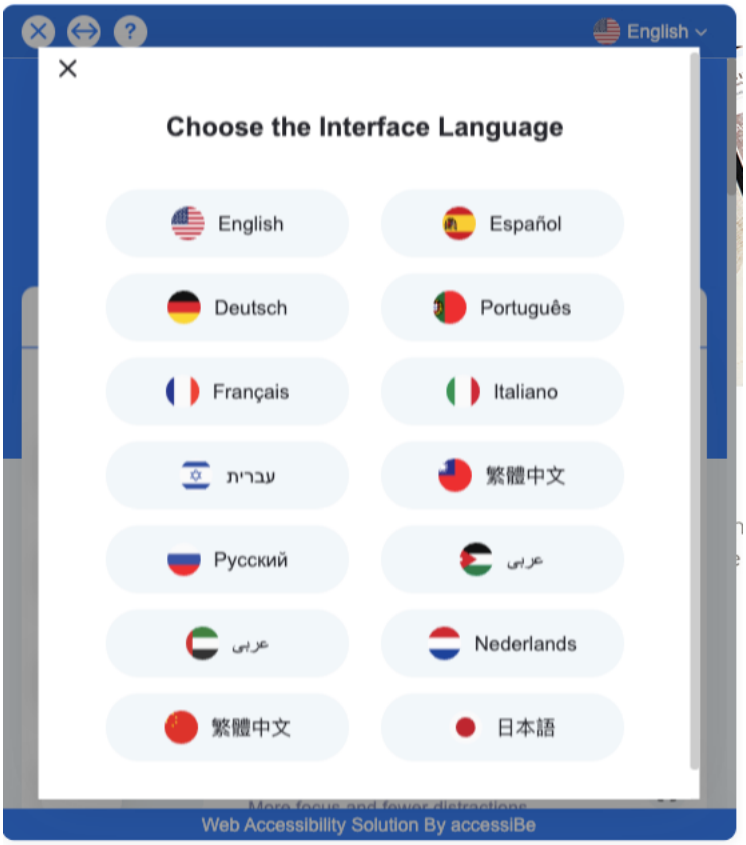


Figure 28: Screenshot of the accessiBe widget's language options

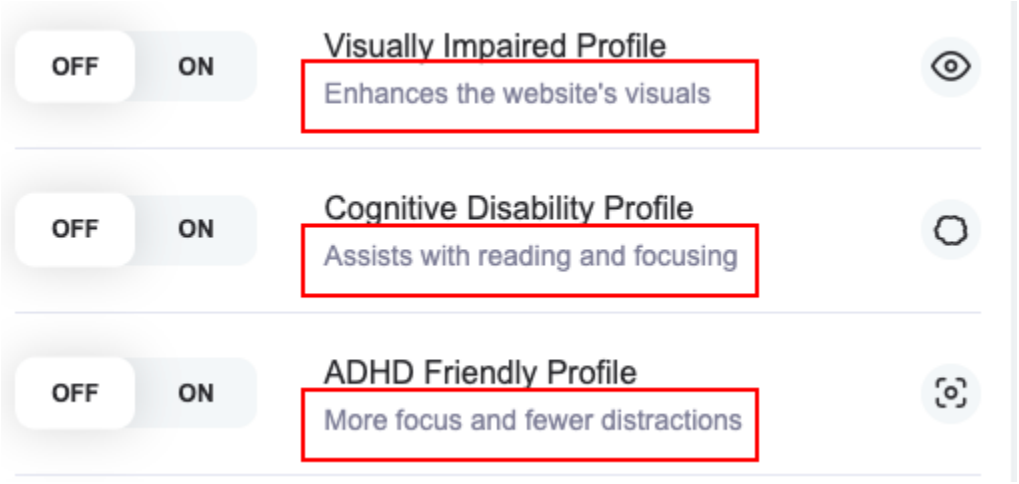
Each of the language options are presented as a button with an icon and the text for each language, localized to the proper name of the language for speakers of same.

As demonstrated in the following code snippet, taken from the button used to select the German language option, the `<img>` element showing the German flag does not have a text-based alternative or a means to cause it to be ignored by assistive technology. As a result, the accessiBe widget violates WCAG 1.1.1. Further, the change to German language for the text label has not been identified, which is a violation of WCAG 3.1.2.

```
<div class="acsb-language" role="button" tabindex="0" data-acsb-
language="de">
    <span class="acsb-language-flag">
        <img
src="https://acsbapp.com/apps/app/assets/media/languages/de.svg">
    </span>
```

```
    <span class="acsb-language-text">Deutsch</span>
</div>
```

Inside the widget's options, the descriptive text underneath each profile type has an insufficient text color contrast of 3.9:1 (should be 4.5:1 per WCAG 1.4.3).



Finally, some operations within the widget will cause the widget to close with no warning provided to the user that such a change of context will occur, which violates WCAG 3.2.2

Conclusion

Sole reliance on accessiBe will not allow a website to achieve full and equal access for users with disabilities.

1. accessiBe's Terms of Service acknowledges that the product cannot bring into compliance any content in Flash, Java, Silverlight, SVG, or PDF.

2. accessiBe's Terms of Service acknowledges that the product cannot bring into compliance any content presented in video and audio.

3. accessiBe's Terms of Service expressly disclaims content that uses invalid markup, which is around 80% of pages on the Web.

4. Testing reveals that accessiBe does not effectively fix problems with
    a. Images and non-text content

b. Document structure
c. Forms
d. Keyboard accessibility and focus control
e. Content within i-frames

5. In addition, the accessiBe widget itself introduces new accessibility issues.

This report demonstrates that implementing the accessiBe product onto a website cannot ensure full and equal access to a website as required by the ADA. As a result, clients must not rely solely on accessiBe and must instead take a more direct and strategic approach to managing their own accessibility.

Appendices

Appendix A: Automatically Tested Websites of accessiBe customers

1. american-apartment-owners-association.org
2. amp-research.com
3. annieselke.com
4. arteza.com
5. belkin.com
6. bensdiscountsupply.com
7. bestbuyautoequipment.com
8. bigdotofhappiness.com
9. bigelowtea.com
10. budgetgolf.com
11. bulkapothecary.com
12. bushwacker.com
13. candlescience.com
14. cariloha.com
15. cases.com
16. conference-board.org
17. decadeawards.com
18. dinntrophy.com
19. eataly.com
20. eset.com
21. glyde.com
22. hdaccessory.com
23. hoveround.com
24. hyperikon.com
25. kiyonna.com
26. knfilters.com
27. linksys.com
28. makingcosmetics.com
29. manhattanportage.com
30. nemoequipment.com
31. neurogan.com
32. oransi.com
33. pinnaclepromotions.com
34. rag-bone.com

35. rampageproducts.com
36. rollnlock.com
37. ross-simons.com
38. sagegoddess.com
39. samys.com
40. scottevest.com
41. seatow.com
42. showmecables.com
43. sidneythomas.com
44. stampedeproducts.com
45. t3micro.com
46. teleflora.com
47. the-cover-store.com
48. tonnopro.com
49. valleyvet.com
50. wenzelco.com

Appendix B: Manually inspected websites of accessiBe customer

1. annieselke.com
2. belkin.com
3. candlescience.com
4. kiyonna.com
5. kiyonna.com
6. scottevest.com
7. showmecables.com
8. the-cover-store.com