Web Accessibility

Annual Report

2020

accessiBe

CONTENTS

Current Environment

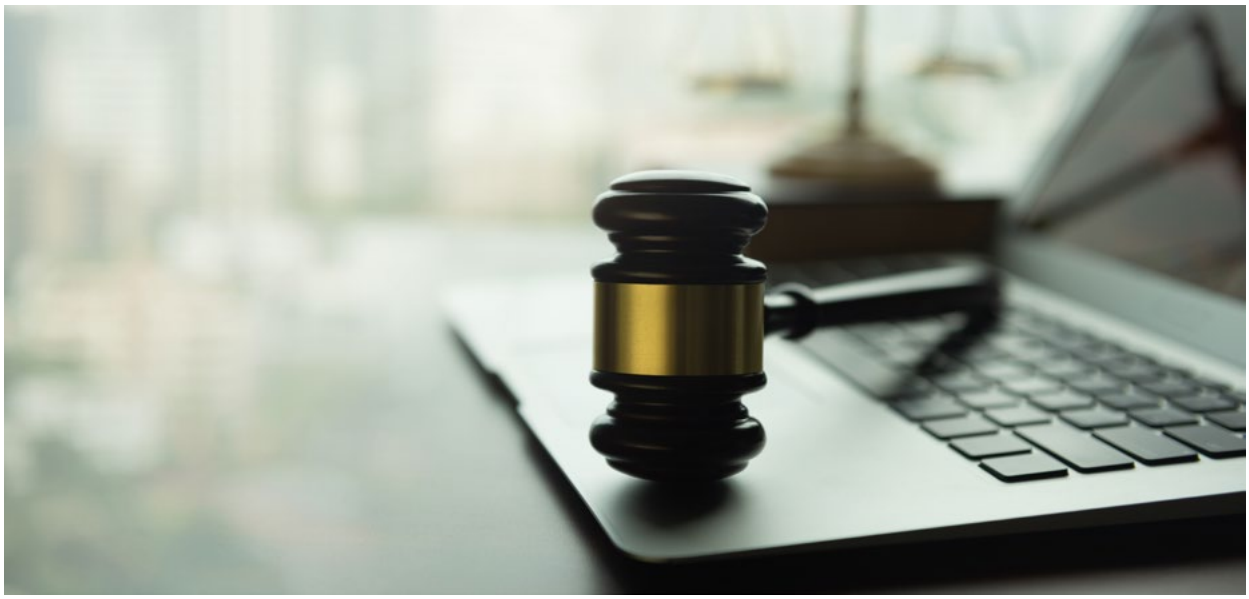
In October 2018, the Department of Justice (DOJ) made it clear that websites are places of public accommodation and, therefore, must comply with the Americans with Disabilities Act (ADA). This rule was validated in January 2019 in the Domino's Pizza web accessibility case.

Since then, tens of thousands of demand letters were issued to small businesses for violations of the ADA. Searches for ADA and the Web Content Accessibility Guidelines or WCAG (the "go-to" guidelines for web accessibility) on Google have risen by more than 400%.

With this in mind, the accessiBe team, decided to check how accessible the internet is. We analyzed more than 10 million web pages and used the WCAG 2.1 AA criteria as the guidelines for our analysis.

Despite a formal DOJ ruling, we discovered that 98% of the scanned web pages in the US failed to pass the WCAG 2.1 compliance requirements. Consequently, this means that **almost all of the websites that we analyzed are exposed to lawsuits.**

This web accessibility report reviews the WCAG 2.1 guidelines and explains where webpages are failing to comply.

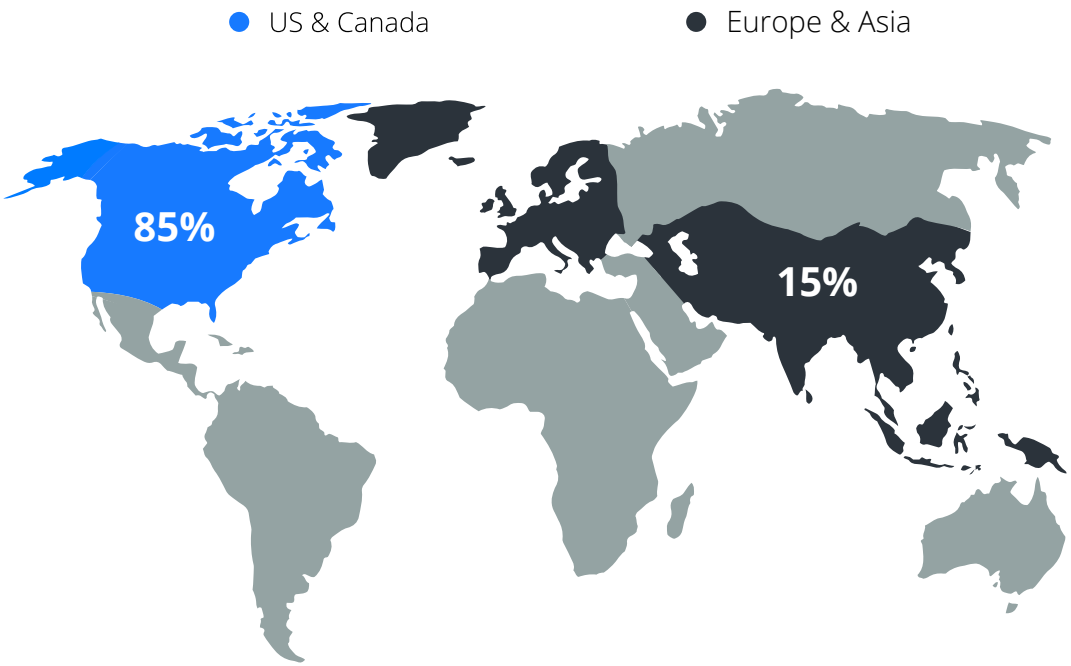


About the Research

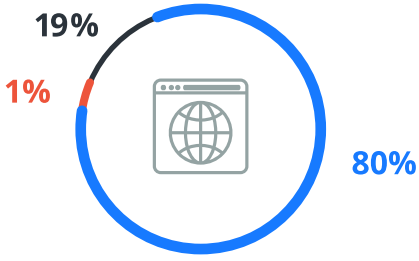
Websites:

We analyzed more than 10M web pages from 50,000 websites.

Website Geolocation



Web Pages per Domain



- Fewer than 1K web pages
- 1K to 100K web pages
- More than 100K web pages

Domain Level

- Top level domains (.com, .net, .org, etc.)
- 2nd and 3rd level domains (.io, .co, .app, etc.)

What We Looked For

Using the WCAG, we checked full website accessibility with the following assistance tools:

A Screen Reader

A Keyboard

Other Assistive Technologies/ Tools

Usable for people with Various Disabilities

If there was a minor, non-repeating guideline error, we did not include it in the results.

For example, if a 'shopping cart' icon did not include an aria-label/text referring to the section for ‘icons’ and an identifiable role, we did count it as accessible.
However, if there were 4 icons, and only 3 of them had an aria-label/text, it passed.

Unlike common automated testing tools such as WAVE, aXe, or Teno, we emulated a browser and used AI to determine element roles.

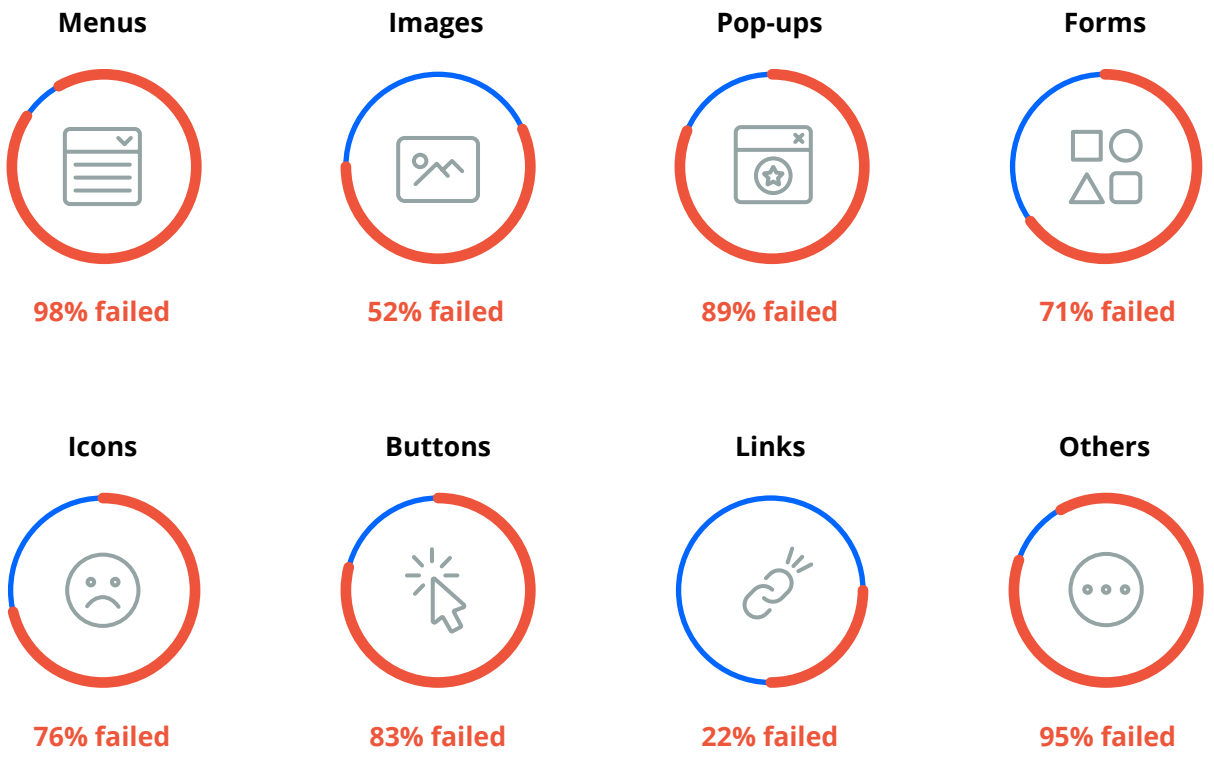
Report Structure

Our report focus on 8 categories within the WCAG 2.1:

1. Menus **2.** Images **3.** Popups **4.** Forms **5.** Icons **6.** Buttons **7.** Links **8.** Others

In each section, we explain the WCAG 2.1 AA guideline and then show where, how, and if the web pages analyzed complied with that specific guideline.

Results Summary:



This diagram shows the percentage of web pages that failed to comply with the minimum guideline requirements for each category, and therefore remain open to lawsuits.

Remember: if any of these guidelines are not adhered to, the entire website becomes non-compliant with the WCAG 2.1 AA.

1.

Compliance Level of

Menus

98% of web pages failed this guideline scan!

WCAG Requirements for Menus

1. A “NAV” tag or a “role” attribute equal to “navigation /menu /menubar” (based on menu type) on the top element that containing all the links and menu items.

2. A “role” attribute equal to “menuitem” on the links that comprise the menu items.

3. Ability to use the *Tab* and *shift+tab* key to navigate between elements.

4. Easily identifiable focused element using a focus ring (outline).

5. Menu bar using the *left-and-right* keyboard arrows. When reaching the end of the menu, and pressing the *forward arrow* key, the navigation should loop back to the first item.

6. Dropdowns menus open using *Enter* and the *down-arrow* keys or by focusing on the menu item.

7. Dropdown navigation using the *up*-and-*down* arrows.

8. Focus remains within the dropdown unless it was intentionally closed using the *Esc* key.

9. The keyboard focus then returns to the root menu item of this dropdown.

Our Findings

✖ Almost none of the websites implemented all menu requirements.

Some enabled navigation using the *Tab* key. Others even took it a step further and implemented *Tab* navigation in dropdowns.

However, few allowed users to close the dropdowns with the *Esc* button, or implemented proper arrow-key navigation. We rarely found proper role attributes.

✔ Some popular templates within the largest content management systems (CMS), such as WordPress, Shopify, Joomla, Magento, BigCommerce, Squarespace, and others, sometimes included built-in accessibility features or plug-ins relating to menus.

The Breakdown

#	Requirement	Passed
1	Implementation of "NAV" tag or role	41%
2	Implementation of role=menuitem	8%
3	*Tab* and *Shift+Tab* navigation including visual focus	12%
4	***Left*-and-*right* arrows menu bar navigation and looping**	<1%
5	Open dropdowns with Focus, *Enter*, and *arrow-down*	5%
6	**Dropdown navigation dropdowns using *up*-and-*down* keys**	<1%
7	Dropdowns can be closed with *Esc* and focus returns to the root item	3%

2.

Compliance Level of Images

52% of the websites failed this guideline test!

WCAG Requirement for Images

Images must have an alt attribute (or alt-tag) that properly describes the objects and the meaning of the image.

If the image contains text, the embedded text must also be present in the alt attribute.

To test for alt attributes, we used both Optical Character Recognition (OCR) technology and IRIS. We looked for words included in the provided alt attributes which described objects and text that appearing in the images.

Our findings

- ✔ Images in 48% of web pages had proper alt attributes.
- ✖ Some websites tried to use shortcuts using file names as the alt attributes. Others were more creative, removing the hyphens from the file names to form what looks like a normal sentence.

  Unfortunately, this doesn't meet requirements.
  Auto-generated strings and other manipulations also did not pass.

3.

Compliance Level of Popups

89% of web pages failed this guideline test!
(Including websites that use popup plugins claiming to be accessible).

WCAG Requirements for Popups

1. When a popup appears, the keyboard focus must lift from the main page and land within the first clickable element of the popup.
2. *Tab* key navigation within the popup must loop back to start after reaching the last element.
3. Popups close using the *Esc* key, and the focus returns to the main page element that was focused on before the popup appeared.
4. Popups must include a "role" attribute equal to "dialog."
5. Popups must include an "aria-modal" attribute equal to "true."

Our Findings

Most websites use some form of a pop-up plugin, either a raw jQuery, embedded within the template like jQuery UI Dialog and Fancy Box, a CMS plugin like OptinMonster, or a third-party service like Zoom Analytics.

Some of these services implement a variety of accessibility features, but most did not. **Those with accessibility features only follow the basic guidelines and do not fully comply with WCAG 2.1 AA.**

The problem with inaccessible popups is two fold:

- When the popup appears, it blocks the screen. Keyboard users may not be able to interact with it.

- Screen reader users may not even know the popup appeared.

A good number of popups implemented *Esc* for closing, but a very small number of them returned the focus to the previously focused element.

The Breakdown

#	Requirement	Passed
1	Keyboard focus enters the popup	18%
2	*Tab* key navigation within the popup	4%
3	Closing the popup with *Esc* **AND** returning the focus to the prior place	2%
4	Popup implementation of role=dialog	31%
5	Popup implementation of aria-modal=true	37%

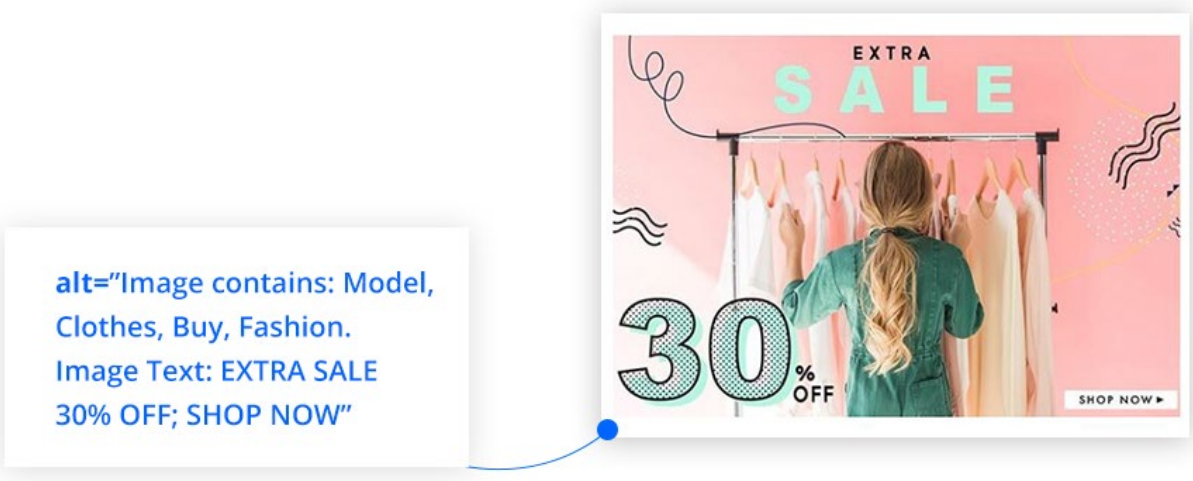
4.

Compliance Level of Forms

71% of web pages failed this guideline test!

WCAG Requirements for Forms

1. All fields must include a “label“ tag connected to the field by the "id" and the "for" attributes, or an "aria-label” attribute.
2. Required fields must include both visual cues (asterisk, text, or other), and the “aria-required” attribute to equal true.
3. Fields must include the “aria-invalid” attribute to inform screen-readers whether the field is currently valid or invalid. This attribute must change dynamically according to the validations. For example an empty required “name” field must include aria- invalid=”true” to indicate that it’s invalid, but change to aria-invalid=”false” once the user fills it out.
4. When a form is submitted, and errors are present, the keyboard focus must be taken to the first invalid field, and the user must receive an explanation (both visual and to the screen reader) of what issue requires addressing.
5. When a form is submitted successfully, a screen reader should be aware of that using an alert element or via other means.



Our Findings

✔ In ready-made, closed systems, like Shopify, Volusion, and BigCommerce, either the platform or its popular form plugin, do a fairly good job in ensuring forms are accessible.

✖ Open-source platforms, such as WordPress, Magento, and Joomla, are open to manual changes and overrides. Although this is good for website managers, this is bad for accessibility. **Why? Because website managers usually don't apply the necessary compliance requirements to changes made manually.**

We also found that even if forms were accessible and most fields were properly described, very few reported a successful submission to a screen reader. Some did report a failed submission, because of built-in HTML5 validations (that are compliant by default).

5.

Compliance Level of Buttons

83% of web pages failed this guideline test!

WCAG Requirements for Buttons

1. Must contain text, title, or an aria-label.

2. Must be an actual “button” tag or a “role” attribute that equals the “button" present.

3. Buttons must include text / aria-label / title.

Our Findings

Most buttons are not tagged properly!

Most websites use “span”, “div” or “a” (link) HTML tags, for creating buttons (without using role=button as a fallback). Developers change how these tags look by default using CSS, and their behavior using JavaScript click events.

Doing this prevents the buttons from being read by screen readers and means they are not clickable using the keyboard.

Almost every website included such non-usable buttons. Standard “button” tags were used almost exclusively with forms, very rarely as standalone buttons, and very few used the “role” attribute as a fallback for screen readers.

6.

Compliance Level of Links

78% passed the guideline test but with a caveat!

WCAG Requirement for Links

1. Must contain text, title, or an aria-label.
2. Must be logically ordered within the document. For example, a “read more” must come after the title and the paragraph of a section.
3. Links must be reachable by keyboard navigation using the *Tab* key.
4. Links must provide a visual indicator if they are opened in a new window.
5. Links must be noticeable on-page and look different than regular text.

Our Findings

Due to default browser behavior, there isn’t much work needed to be done for link compliance. Most websites passed the links test.

Having said that very few of the links indicated a new window being opened. We passed websites anyway because this is not a big compliance issue. If we did grade websites links to this standard, we’d have to fail almost all of them, despite being usable to most site visitors with screen readers.

Additionally, there were many links that did not contain text, titles, or aria-labels, but just an icon, such as social media links. In our analysis, we did not address these as links but as icons, which are addressed in the next section.

7.

Compliance Level of Icons

76% of web pages failed this guideline test!

WCAG Requirements for Icons

Since there isn't a tag or an element for icons, there aren't any guidelines. Icons are usually used with links or buttons to describe an action. Therefore, the requirements for making icons accessible are the same as making links and buttons compliant.

Why do icons need to be accessible? Icons appear in almost 100% of websites, and even though they don't have their own functionality, developers use them differently than buttons and links.

To explain this, we first need to establish why icons are most often used for. Icons usually exist as an image of a button or a link and aim to describe the link.

For example, many eCommerce websites use a shopping cart icon without any text explaining that when clicked, the shopping cart will open.

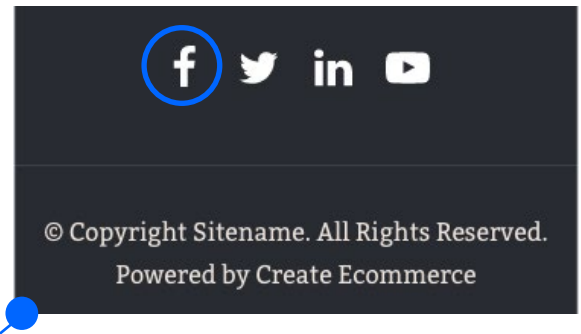
Other examples are social media logos without text, favorites or wishlists often with a heart icon, search buttons with a magnifying glass, product ratings (star icons), etc.

Our Findings

Most icons are “SVG” elements, font icons (like font awesome or IcoMoon), or background-images (as Sprite or standalone). Neither of these practices is helpful for people using screen readers.

To make these icons accessible, developers have to include a hidden text or an aria-label attribute, describing what this icon is. For example, a Facebook icon should have to have a hidden text or an aria-label equal to “Facebook.”

We found that most icons are not being described, especially font-icons.
Still, 24% of the web pages did pass the test, due to some popular icon libraries implementing descriptions built into their SVG elements. Other font-icon CMS plugins also do something similar but on a different type of element (usually an “I” tag).




```
<a href="https://www.facebook.com"
title="facebook|NewWindow" target="blank"
class="Icon-model-social-networks facebook-menu-root-
link"rel="noreferrer"data-acsb-clickable="true"
data-acsb-currently-navigable="true"
aria-label="facebook"
data-acsb-aria=label=method="social"></a>
```

8. Other Findings

There are other important accessibility requirements websites must include to be compliant that don't belong in any of the above categories.

HTML language attribute: The "HTML" tag must include a "lang" attribute equal to the language of the document. For example, "EN" for English and "FR" for French.

Skip links: The top of every web page needs to include "visually hidden" buttons (that appear on focus with a screen reader) and allow users to skip to different sections within the page.

Incorrect use of aria: Many websites include faulty usage of aria attributes, usually faulty "aria-labelledby" and "aria-describedby" that are automatically populated by various forms of plugins.

Tabbing in hidden elements: Web pages often include hidden elements that appear in certain conditions such as popups, sidebars, and dropdowns. These elements must not be navigable to screen readers when not visually active on the screen.

Weak or non-existent focus ring: Every clickable element must include a visual, noticeable focus-ring (outline) when being focused during *Tab* navigation.

#	Requirement	Passed
1	Proper HTML "lang" attribute	65%
2	Skip links	18%
3	Incorrect use of aria attribute	15%
4	Tabbing in hidden elements	39%
5	Weak or non-existent focus ring	32%

Our Takeaways

The internet is in very poor compliance shape today, restricting millions of people from full or partial use.

Through the research, we see that plugin and template developers do try to follow certain WCAG guidelines, but their efforts are often insufficient.

Moreover, these plugins and templates are being used by website owners.
When changes are made to plugins or templates, they may compromise the level of accessibility for the entire website if they aren't aware of accessibility guidelines.

Remember: accessibility is difficult to achieve manually.

About accessiBe

The first and only AI-powered and
fully automated solution.

accessiBe was founded by 3 entrepreneurs who were looking for a web accessibility solution for their clients. They soon discovered that the available solutions were manual, complicated, cost tens of thousands of dollars and took too long to implement.

They set out to develop an affordable and easy-to-use tool that any website owner can use to avoid unnecessary legal action.

Harnessing the power of artificial intelligence, they created accessiBe, the first and only fully automated solution for web accessibility.

accessiBe uses state-of-the-art machine learning technology to optimize and adjust any website to the strict standards of the WCAG 2.1 AA, which are the most accepted guidelines for web accessibility worldwide.

To create the best solution, the 3 entrepreneurs didn't settle just with the standards set out in the WCAG, but also partnered with people with disabilities to learn from them how an accessible website should look and feel.

Among other web accessibility changes, accessiBe provides an interface for people with disabilities. This allows them to adjust the website UI, to fit the look and feel of the site to their needs without altering the source code.

It takes accessiBe just 48-hours to make an entire website fully accessible with only one line of code and an easy 5-minute installation.

accessiBe's solution allows every website owner to comply with any WCAG-guided accessibility law in the world. accessiBe's clients are fully compliant with regulations in the US (section 508 of the Rehabilitation Act and the ADA), Canada (the ACA), the EU (EAA/EN 301549), and more.

This has led to over 10,000 website owners trusting accessiBe; from small business owners to Fortune 500 companies.

Most importantly, accessiBe set the goal of making the entire digital world fully accessible for people with disabilities by 2025.

Contact us **now** to learn how you can make **your** website accessible.