

How to test your web app accessibility?

by Sylvain Hamann

What's Web Accessibility?

It's the principle of designing and developing websites in a way that ensures they can be used by anyone, regardless of their abilities or disabilities, and regardless of the technology they use to access the web.

MANUAL AUDIT

Keyboard only navigation

Via tab or shift+tab

A screen reader is a form of assistive technology (AT)^[1] that renders text and image content as speech or braille. Screen readers are essential to people who are blind,^[2] and are useful to people who are visually impaired, illiterate, or have a learning disability.^[3] Screen readers are software applications that attempt to convey what people with normal eyesight see on a display to their users via non-visual means like text-to-speech,^[4] sound icons,^[5] or a braille device.^[2] They do this by applying a wide variety of techniques that include, for example, interacting with dedicated accessibility APIs, using various operating system features (like inter-process communication and querying user interface properties), and employing hooking techniques.^[6]

All Microsoft Windows operating systems have included the Microsoft Narrator screen reader since Windows 2000.

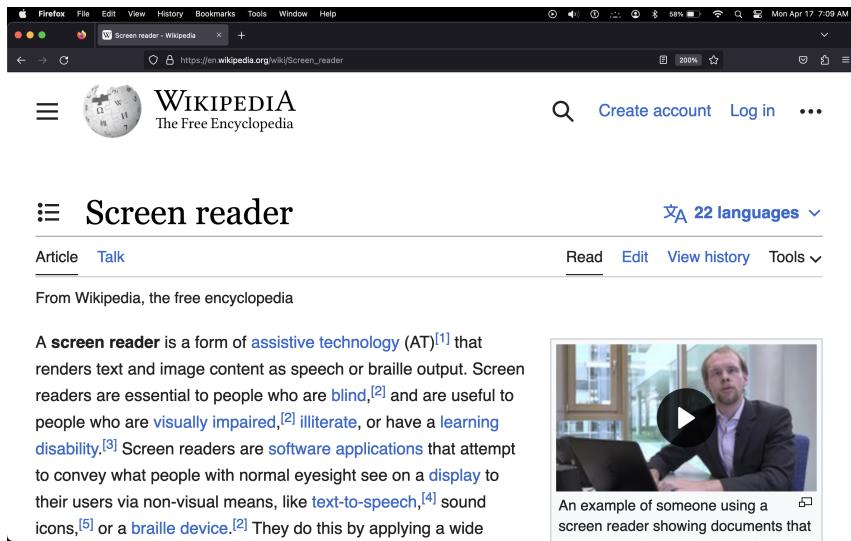


An example of someone using a screen reader showing documents that

Zoom

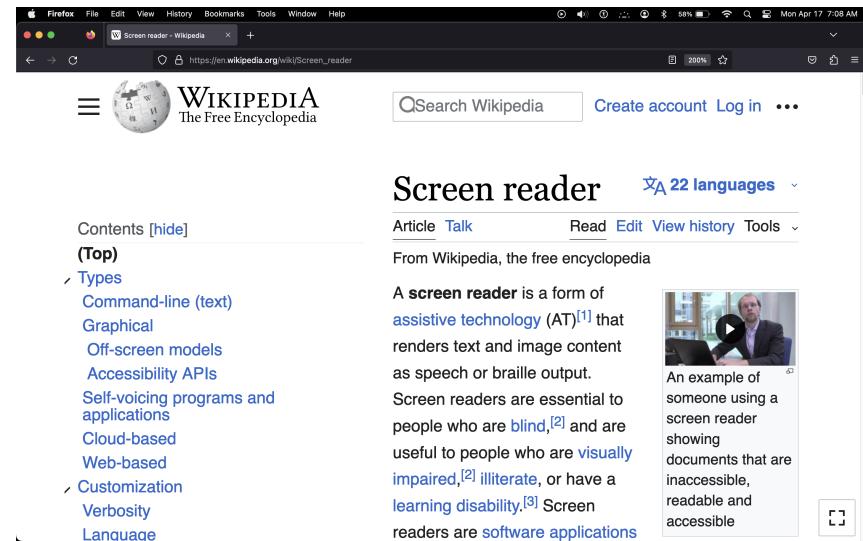
- Try different types of zoom and magnifiers;
- WCAG recommends to support 200% zoom.

Default zoom



A screenshot of a Firefox browser window displaying the Wikipedia article "Screen reader". The page content is visible at 100% zoom. The browser interface includes a top menu bar with options like File, Edit, View, History, Bookmarks, Tools, Window, Help, and a zoom level indicator of 100%. Below the menu is a toolbar with back, forward, search, and other navigation buttons. The main content area shows the Wikipedia logo, the title "Screen reader", and the beginning of the article text. A video player is embedded on the page, showing a man using a screen reader.

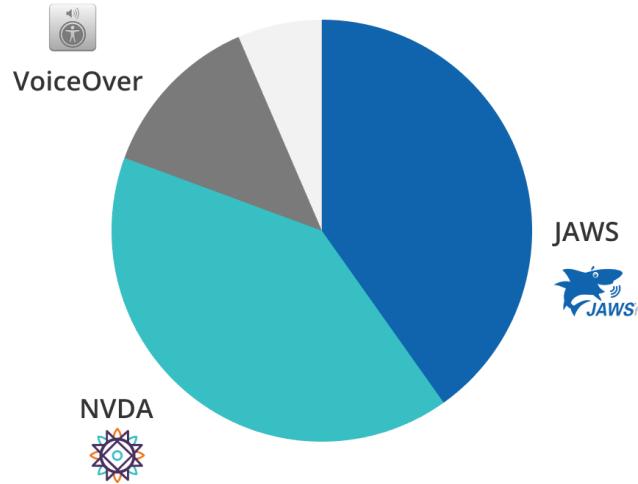
Text only zoom



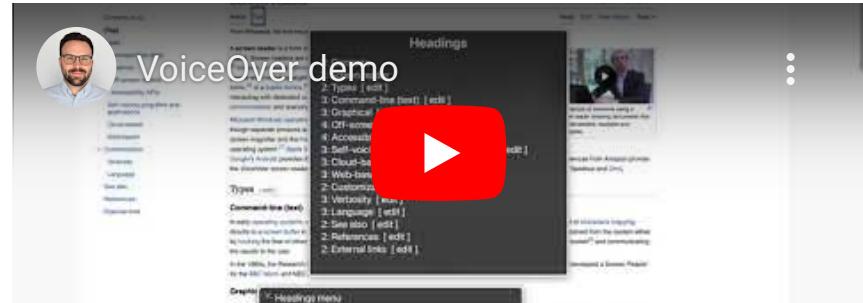
A screenshot of a Firefox browser window displaying the same Wikipedia article "Screen reader" at 200% text-only zoom. The text is larger and more prominent, while the overall page layout remains similar to the default zoom view. The browser interface is identical to the first screenshot, with the zoom level set to 200%.

Screen Readers

Most popular Screen Readers



VoiceOver on MacOS



source: assistivlabs.com

Responsive Design

- Mobiles and Tablets also have screen readers!
- We saw that zooming might trigger "mobile" layout;
- Each major browser has a Responsive Design Mode.

The screenshot shows a Safari browser window displaying the Wikipedia article for "Screen reader". The window title is "en.m.wikipedia.org". The header includes a shield icon, a refresh button, and the URL. To the right of the URL are zoom controls (414 x 736 (100%) | 3x) and a note (Safari — iOS 16.0 — iPhone). Below the header is a row of icons representing various Apple devices and their screen sizes: iPhone SE, iPhone 8, iPhone 8 Plus, iPad mini (7.9"), iPad (9.7"), iPad Pro (10.5"), iPad Pro (12.9"), 800 x 600, 1366 x 768, and 1920 x 1080. The main content area shows the Wikipedia article page with the title "Screen reader", sections like "Article" and "Talk", and a "This article needs additional citations for verification" notice. The text describes what a screen reader is, its uses, and how it works. At the bottom of the page is a video thumbnail showing a person using a screen reader. The sidebar on the right contains a section about Microsoft Windows operating systems and their screen reader support.

Window Help

en.m.wikipedia.org

414 x 736 (100%) | 3x | Safari — iOS 16.0 — iPhone

iPhone SE iPhone 8 iPhone 8 Plus iPad mini (7.9") iPad (9.7") iPad Pro (10.5") iPad Pro (12.9") 800 x 600 1366 x 768 1920 x 1080

WIKIPEDIA

Screen reader

Article Talk

This article needs additional citations for verification. July 2017 Learn more

A screen reader is a form of assistive technology (AT)^[1] that renders text and image content as speech or braille output. Screen readers are essential to people who are blind,^[2] and are useful to people who are visually impaired,^[2] illiterate, or have a learning disability.^[3] Screen readers are software applications that attempt to convey what people with normal eyesight see on a display to their users via non-visual means, like text-to-speech,^[4] sound icons,^[5] or a braille device.^[2] They do this by applying a wide variety of techniques that include, for example, interacting with dedicated accessibility APIs, using various operating system features (like inter-process communication and querying user interface properties), and employing hooking techniques.^[6]

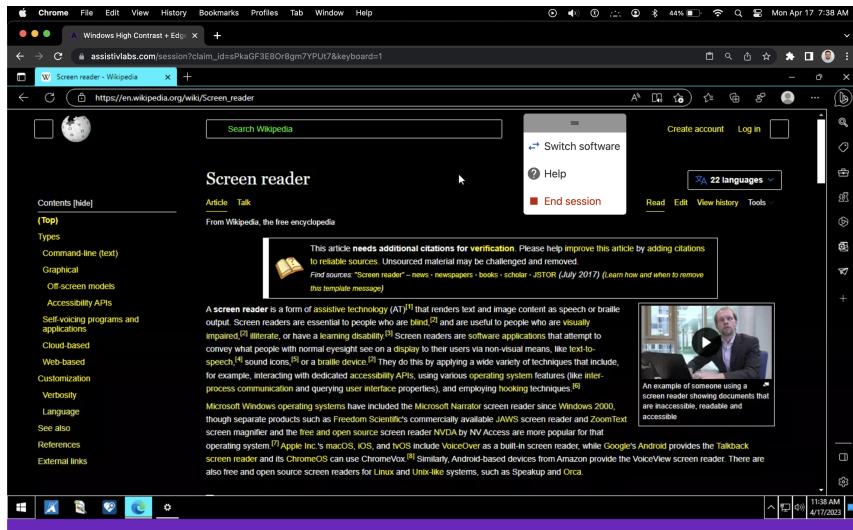
An example of someone using a screen reader showing documents that are inaccessible, readable and accessible

Microsoft Windows operating systems have included the Microsoft Narrator screen reader since Windows 2000, though separate products such as Freedom Scientific's commercially available JAWS screen reader and ZoomText screen magnifier

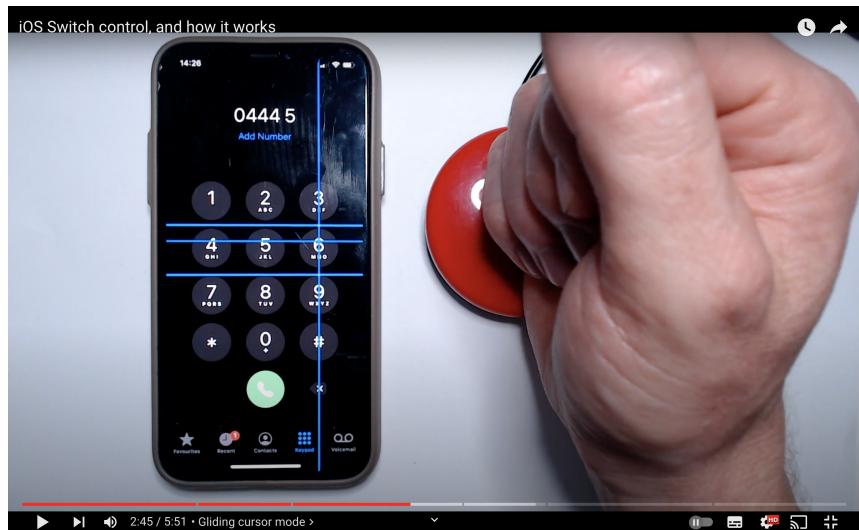
There are many others assistive technologies

For example:

High Contrast Mode on Windows



Switch device



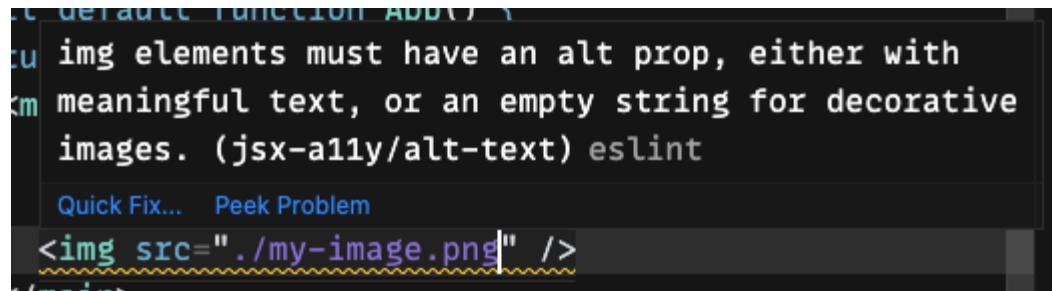
Another one: Reduce screen motion on Mac OS.

Source: Assistive Tech Stuff on YouTube

AUTOMATIC TESTS

Static testing

- ESLint (see `eslint-plugin-jsx-a11y`) can help you to detect a11y issues while coding;



A screenshot of a code editor showing an ESLint warning for an `img` element. The code is:

```
function Add() {
  img elements must have an alt prop, either with
  meaningful text, or an empty string for decorative
  images. (jsx-a11y/alt-text) eslint
  Quick Fix... Peek Problem

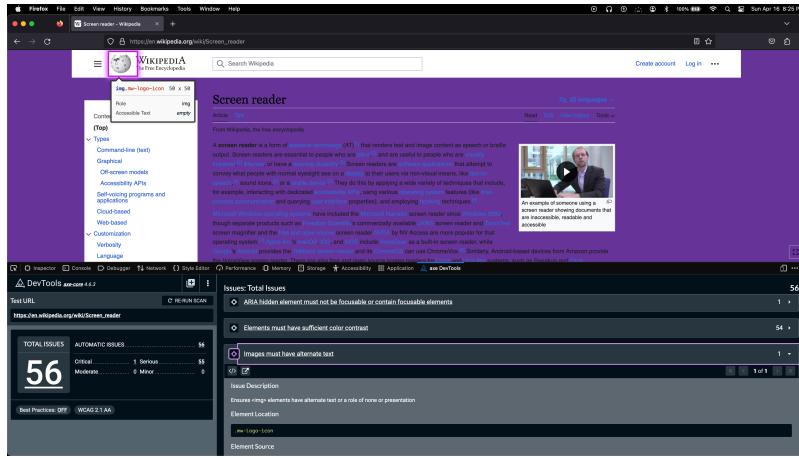
```

The word `img` is underlined with a red squiggly line, indicating a syntax error. A tooltip above the code provides the ESLint rule detail: "img elements must have an alt prop, either with meaningful text, or an empty string for decorative images. (jsx-a11y/alt-text) eslint". Below the code, there are "Quick Fix..." and "Peek Problem" buttons.

- You can create custom rules for your Design System;
- TypeScript plus JSDoc could be used to achieve similar purpose;
- This does not test the DOM.

axe by Deque Systems

Via a browser extension (axe DevTools)



- Lighthouse has a similar feature;
- This is still kind of manual.

Via React Testing Library

```
import { render } from "@testing-library/react";
import { axe } from "jest-axe";

import { Button } from "./Button";

test("Button a11y", async () => {
  const { container } = render(<Button>Click me</Button>);
  const results = await axe(container);

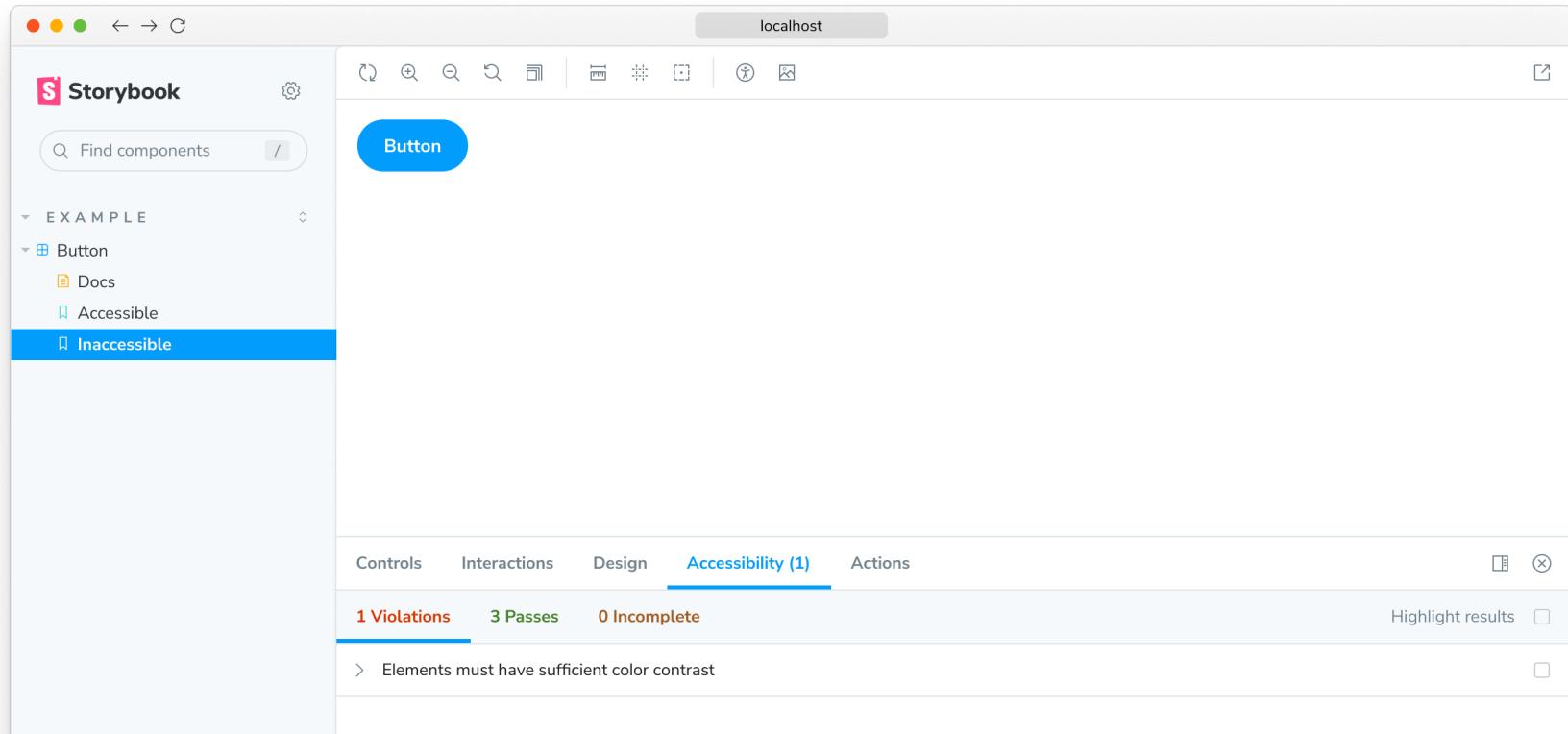
  expect(results).toHaveNoViolations();
});
```

- ⚠️ jsdom won't catch contrast issues;
- Also available for E2E testing libraries such as Cypress or Playwright;
- It feels redundant.

axe can "only" catch up to 57% of a11y issues.

A11y tests with Storybook

- Storybook is a tool for developing, documenting, and testing UI components in isolation;
- Use `storybook-addon-a11y` to check the a11y of each story.



Test keyboard interactions

```
test('Menu keyboard interaction', async () => {
  render(<Menu />);
  const button = screen.getByRole("button", { name: "Dashboard" });

  expect(button).toHaveAttribute("aria-expanded", "false");

  await userEvent.keyboard("{tab}");
  await userEvent.keyboard("{enter}");

  expect(button).toHaveAttribute("aria-expanded", "true");
  expect(screen.getByRole("menuitem", { name: "Profile" })).toHaveFocus();

  await userEvent.keyboard("{arrowdown}");

  expect(screen.getByRole("menuitem", { name: "My account" })).toHaveFocus();

  await userEvent.keyboard("{Escape}");

  expect(button).toHaveFocus();
  expect(button).toHaveAttribute("aria-expanded", "false");
});
```

DASHBOARD

Resources

- [Accessibility Myths](#)
- [Learn Accessibility](#)
- [A11ycasts with Rob Dodson](#)
- [Testing Accessibility by Marcy Sutton](#)
- [Practical Accessibility with Sara Soueidan](#)

Conclusion

- Acknowledge your personal biases;
- Include a11y audits during design phase, code reviews, bug hunts...
- Consider all possible interactions in your automatic tests;
- Seek feedback from individuals with diverse abilities via companies such as Fable or by hiring them!