Engineer

Accessibility

a site's fitness for people with disabilities

What it is

Web accessibility is how well a website, an app or a tool works for people with disabilities. The W3C's Web Accessibility Initiative (WAI) defines it by what people can do: "perceive, understand, navigate, and interact with the Web", and contribute to it. The last verb carries weight: a site where a blind visitor can read every article but cannot post a comment, upload a file or finish a sign-up has met half the definition. The usual shorthand, a11y, counts the eleven letters between the a and the y.

The scope is every disability that affects access to the web. WAI lists auditory, cognitive, neurological, physical, speech and visual, and each brings its own barriers: video without captions, instructions that lean on memory, motion that triggers symptoms, controls that need a steady hand and a mouse, voice interfaces that expect fluent speech, text that works at one size and one contrast only.

Underneath the visible page, the browser builds a second structure for assistive technology. The ARIA specification calls it the accessibility tree: a tree of accessible objects in which each node represents an element of the interface, such as a push button or a check box, exposed through the platform's accessibility API. Screen readers, voice control, switch devices and braille displays work from that tree rather than from pixels. Each node carries a name, a role and a state, and under WCAG's criterion 4.1.2 Name, Role, Value, software must be able to determine them for every interface component. Native HTML controls, the standard notes, already meet 4.1.2 when used according to specification. A `div` with a click handler arrives in the tree as generic text, with no role and no place in the keyboard order.

Accessibility depends on several parts working together, WAI notes: web technologies, browsers, authoring tools and the websites themselves. A site owner controls the last of these, which is where WCAG sets out in testable terms what content has to do.

Use the concept in a decision

Put “What is accessibility?” to work in a real decision. We can help relate the concept to the system you are building or buying. Layout, wording and flow arranged so people finish what they came to do instead of ringing your support line.

Explore our services Get in touch →

Test the assumptions behind the term

Use the explanation and linked work to frame questions for your project. We can examine the assumptions with you before a tool is chosen. Presented 200 UI/UX improvements — 200 interface improvements shipped to the map

Presented 200 UI/UX improvements Get in touch →

Why it matters

The web carries education, work, government services, shopping and health care, and WAI points out that the UN Convention on the Rights of Persons with Disabilities defines access to information and communications technology, the web included, as a basic human right. Accessibility is also required by law in many situations, and buyers turn that into a contract line naming WCAG 2.2 Level AA. That line decides what the supplier has to build and prove.

The same work serves people who would never call themselves disabled. WAI’s own list includes people on phones, watches and televisions with small screens and different input modes, older people whose abilities change with age, someone with a broken arm or lost glasses, someone in bright sunlight or in a place where audio cannot be played, and anyone on a slow or expensive connection. Captions help on a silent train, and strong contrast helps on a phone at noon.

Timing sets the price. WAI calls it most efficient to build accessibility in from the very start of a project, so that work does not have to be redone. Choosing a native button costs nothing on the first day. Replacing a hand-built one after launch means fixing the component, then re-testing every page that used it, in every language and theme the site ships.

Evaluation follows the same logic. Tools help, but WAI is plain that no tool alone can determine whether a site meets accessibility guidelines and that knowledgeable human evaluation is required. A rule engine reads what the

markup says. A person with a keyboard and a screen reader finds out what the page does.

What it looks like in practice

Eleven defects survived on a site that passed its automated accessibility rules, because each was a property of how the page rendered rather than of what the markup said, and fixing eleven defects the browser called healthy started from that gap. Eight footer links stayed in the tab order behind a property that removes pointer interaction and nothing else, so a keyboard could land on links that were invisible on screen. Two defects sat in the guards themselves: every window the layout check opened was nine hundred pixels tall, so on a landscape phone at 852 by 393 two items sat seven pixels apart with nobody looking, and the rule engine had been running seventy of its hundred and five rules. Four rules came out of the work, among them that a guard testing one axis defends one axis and that visibility is proved in pixels rather than in the document. Reopened properly, the layout check failed a hundred and twenty-eight times across three languages.

An application built with SwiftUI needed the same work in another form, naming all seventeen icon-only controls. A system symbol with no accessibility label is read out by its internal name, so the send button announced itself as “arrow up circle, button”, and every other icon-only control did the same in its own way: seventeen in all described their artwork instead of their function. Each now has a label that names the action, and where the meaning depends on state, the label follows the state. A linter requires a label within eight lines of any construction that creates an icon-only control, a deliberately crude rule chosen by measurement, so an eighteenth control cannot be added without one. The rule has a limit of its own: it proves a label exists nearby, and whether the label is right still takes someone listening.

The same structure now serves a newer reader too, the subject of readiness for AI agents. Many agents that work a browser read a page the way a screen reader does, through its accessibility tree, so the real buttons, labelled forms, headings in order and landmarks that make a site usable without a mouse are the same work that lets an agent find a form and fill it in.

Where it goes wrong

A custom dropdown works with a mouse and traps everyone else. A `div` with a click handler has no role, takes no keyboard focus and reports no open or closed state, so a screen reader announces plain text and the Tab key passes it by. Start from the native element, `select`, `button` or `details`, which brings role, focus and state with it. When a design truly needs a custom widget, follow its pattern in the ARIA Authoring Practices Guide, then test it with a keyboard and a screen reader.

Alt text describes the picture when the reader needed its job. 1.1.1 Non-text Content asks for a text alternative that serves the same purpose, and a linked logo described as “blue hexagon logo” tells a screen reader user nothing about where the link goes. Write the alternative for the image’s role: the destination for a linked image, the finding for a chart, the words for an image of text. Give purely decorative images an empty `alt=""` so assistive technology passes over them.

Focus drops to the top of the page after a dialog closes. Closing a dialog or changing route in a single-page app removes the element that had focus, focus falls back to the document body, and a keyboard or screen reader user loses their place. Return focus to the control that opened the dialog. After a client-side route change, move focus to the new view’s main heading, given `tabindex="-1"` so it can take focus, and the screen reader announces where the user has arrived.

Errors appear only as a red border. 1.4.1 Use of Color requires that colour is not the only visual means of conveying information, and 3.3.1 Error Identification asks that a detected input error is identified and described to the user in text. A red outline does neither for someone who cannot tell red from grey or cannot see the field at all. Put a short message beside the field that says what is wrong and how to fix it, connect it with `aria-describedby` so a screen reader reads both together, and keep the colour as a second signal.

Further reading

- Introduction to Web Accessibility — WAI’s definition, who benefits, and why evaluation needs people as well as tools.
- How People with Disabilities Use the Web — stories of users, the barriers they meet and the tools and techniques they rely on.

- The Business Case for Digital Accessibility — reach, innovation, brand and legal risk, set out for the people who approve budgets.
- Evaluating Web Accessibility Overview — from easy checks to a full conformance evaluation, with tools and reporting.
- ARIA Authoring Practices Guide — patterns for custom widgets, with the roles, states and keyboard behaviour each one needs.

Glossary

Choose for your actual constraints

Describe the workflow, the existing interfaces and the users who need the change. We can define a testable delivery and a handover that lets your team maintain it.

See what it costs Get in touch →

Make the terminology practical

Ask how “What is accessibility?” applies to your organisation. A concrete example from your current system is enough to start the discussion.

Explore our services

welcome@engineer.company +45 50 10 61 84 Copenhagen, Denmark

Take this page with you

Contact card

Carefully Crafted With Love by D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Copyright © 2025 – present · Engineer ApS · All rights reserved.

Printed from https://engineer.company/glossary/accessibility/