How Digital Accessibility Works: A Visual Introduction

- See how people with disabilities actually interact with digital products
- Follow a request from person, through assistive technology, browser, and content
- Legal exposure under WCAG 2.2 and the European Accessibility Act
- Broader usability and inclusive quality for design, code, and content teams
- Roadmap: pipeline and accessibility tree, POUR, law, practice, testing, habits

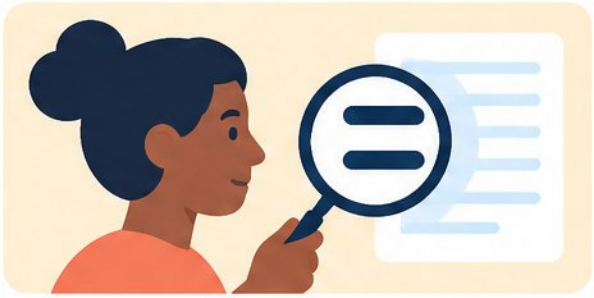


Who Accessibility Serves: Disabilities, Situations, and Assistive Technology

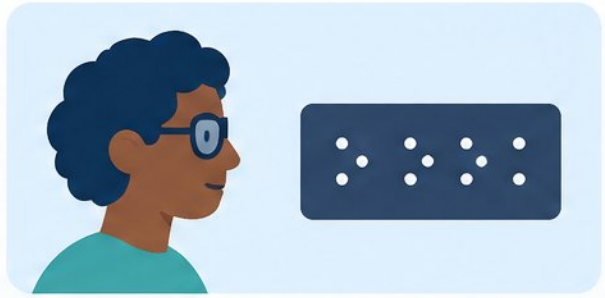
- Disability types: vision, hearing, motor, cognitive, speech — plus temporary and situational limits
- Assistive tech is software and hardware: screen readers, magnifiers, braille displays
- Also voice control, switches, eye trackers, and keyboard-only use
- Same task, different paths: speech, 400% magnification, or captions
- Design for the input and output methods people actually have



Screen readers



Magnifiers



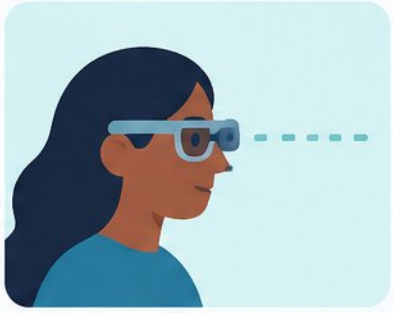
Braille displays



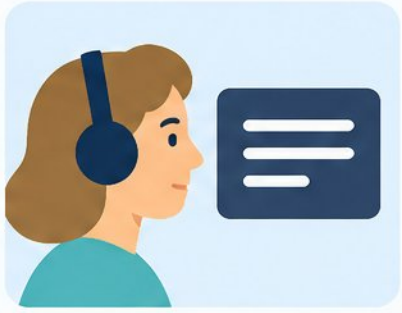
Voice control



Switches

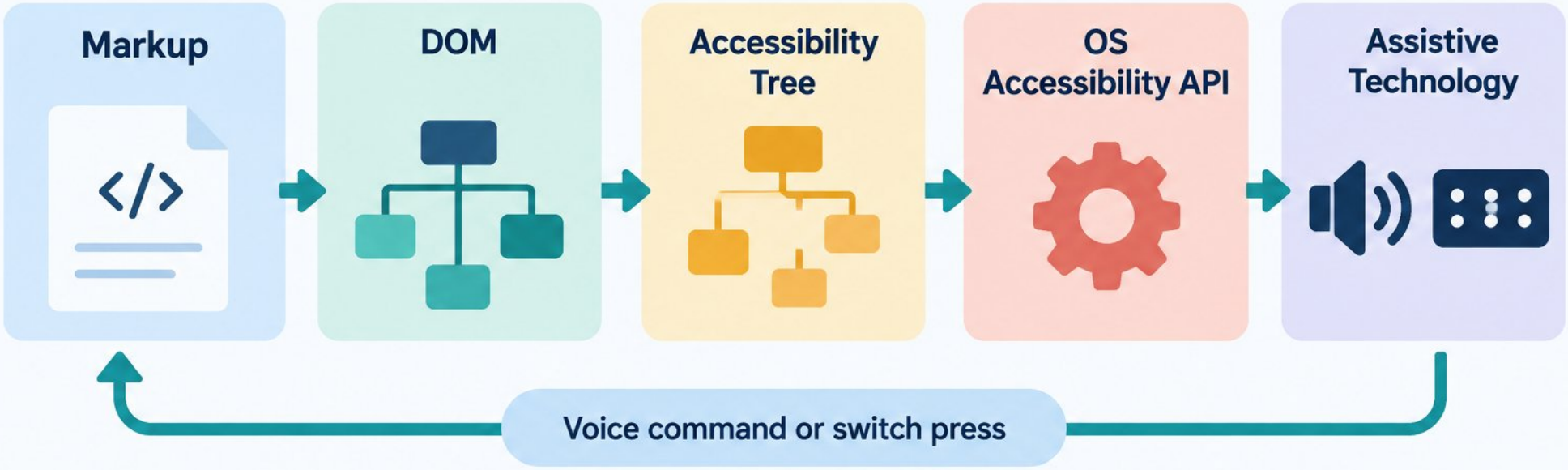


Eye trackers



Captions

The Accessibility Pipeline: From Markup to Speech



- Markup becomes the DOM; the browser builds an accessibility tree from it
- The OS accessibility API exposes that tree to assistive technology
- Every accessible object carries four things: role, name, description, states and properties
- The tree is not the DOM: presentational elements drop out, complex components add objects
- Interaction flows back: a voice command or switch press reaches the browser through the API
- Inspect it yourself with Chrome DevTools or Firefox Accessibility Inspector



Where the Pipeline Breaks: Common Barriers Mapped to Roles

Content failures:
missing or meaningless alt text, vague links like “click here”, uncaptioned video

Design failures:
low contrast, color-only cues, tiny targets, removed focus indicators

Code failures:
divs used as buttons, unlabeled fields, unannounced updates

Interaction failures:
keyboard traps, lost focus, drag-only actions, unextendable timeouts



Role mapping:
authors own content, designers own visuals, engineers own code, PMs own priorities

Authors
own content

Designers
own visuals

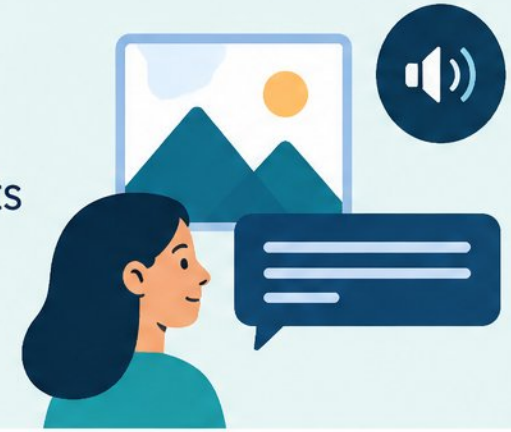
Engineers
own code

PMs
own priorities

The Four Principles: POUR in Practice

Perceivable:

alt text, captions, contrast, and layouts that adapt



Operable:

keyboard access, logical focus order, no traps, adjustable time limits



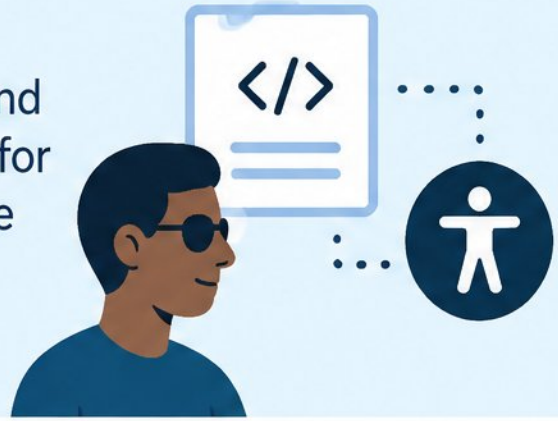
Understandable:

plain language, predictable navigation, clear error messages and suggestions



Robust:

valid semantics and name, role, value for current and future assistive tech





POUR is the why under the checklist — each principle maps to real people

What Changed in WCAG 2.2: The New Success Criteria

WCAG 2.2 extends 2.1 without deprecating it; published 5 Oct 2023, updated 12 Dec 2024

Nine new success criteria since 2.1; 4.1.1 Parsing removed ass obsolete

2.4.11 Focus Not Obscured (AA): focused elements stay clear of sticky headers and footers

2.5.7 Dragging Movements and 2.5.8 Target Size (AA, 24×24 CSS pixels)

3.2.6 Consistent Help and 3.3.7 Redundant Entry (A) reduce memory and hunting

Conformance levels A, AA, AAA: AAA is not a blanket policy for whole sites



Standards and Law: WCAG, EN 301 549, and the European Accessibility Act

Building a more inclusive digital world

WCAG 2.2
also published as **ISO/IEC 40500:2025**, the global benchmark

→

EN 301 549 v3.2.1
still the legal reference

⇢

EN 301 549 v4.1.1
lands 16 December 2026

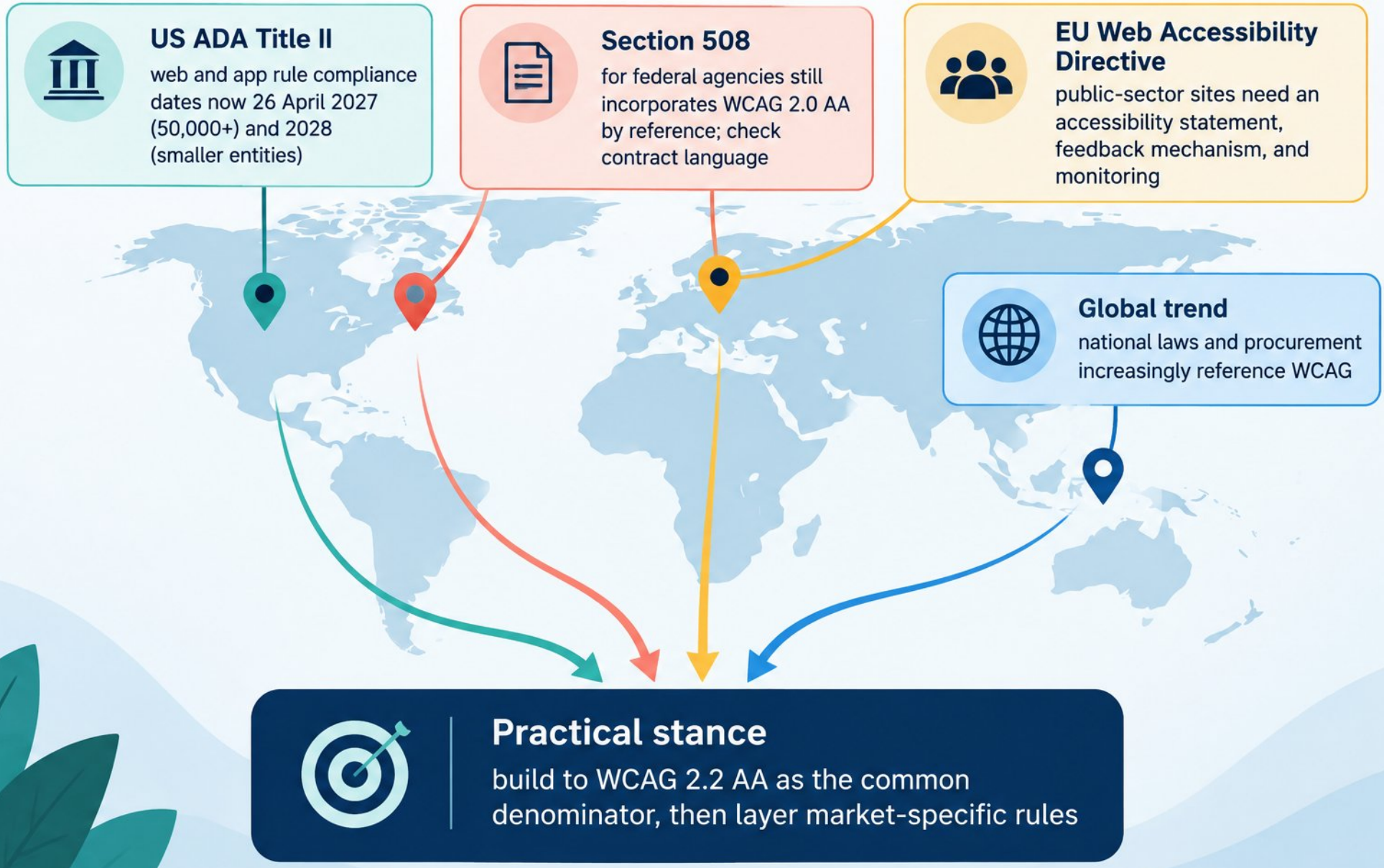
→

European Accessibility Act
applies since 28 June 2025

- WCAG 2.2 also published as ISO/IEC 40500:2025, the global benchmark
- EAA applies since 28 June 2025, including to non-EU companies trading in the EU
- EAA cites EN 301 549, not WCAG directly: v4.1.1 lands 16 December 2026
- Until cited, v3.2.1 (WCAG 2.1) remains the legal reference
- Practice now: audit against WCAG 2.2 AA and document your claim
- Microenterprises under 10 staff and €2M turnover are exempt
- A conformance claim is not a certificate; a scan score is not conformance

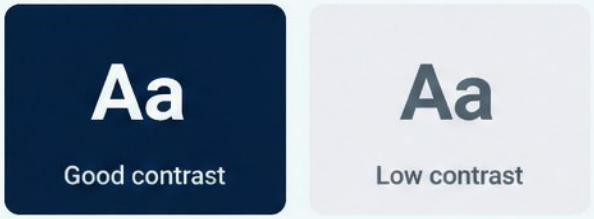


Other Legal Frameworks to Know



Designing for Accessibility: Visual and Interaction Patterns

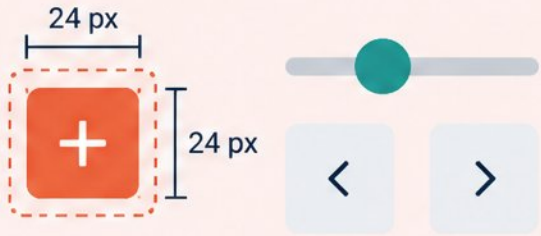
Contrast: 4.5:1 body text, 3:1 large text; never color alone as the only cue



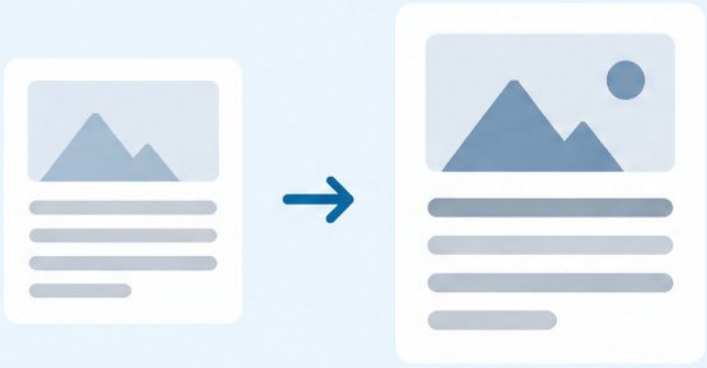
Focus: visible indicator that survives dark, light, and busy backgrounds



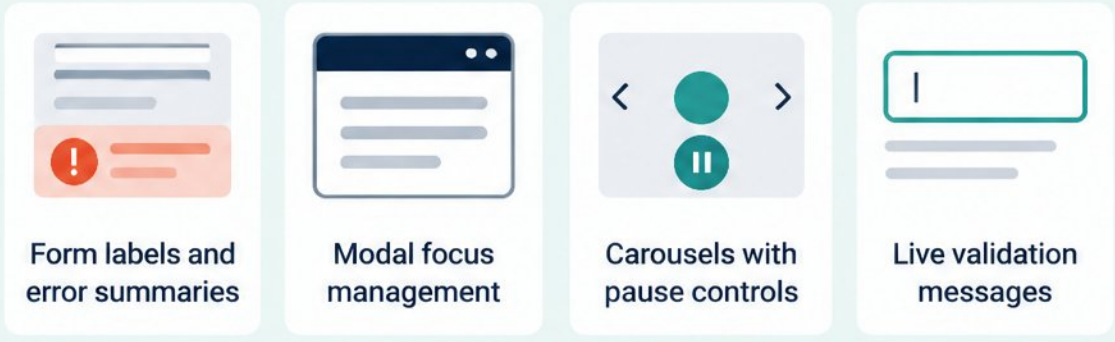
Targets: 24×24 CSS pixels minimum, generous spacing, non-drag alternative for sliders and reorders



Zoom and reflow: survive 200% text resize and 400% zoom without horizontal scrolling



Annotate in specs: form labels and error summaries, modal focus management, carousels with pause controls, live validation messages



Content · Design · Code · Product ownership

Building It in Code: Semantics, Names, and Dynamic Content

HTML (Semantic Markup)

```
<h1>Account settings</h1>

<button type="button">
  Save changes
</button>

<label for="email">
  Email address
</label>
<input type="email" id="email"
  name="email">
```

Accessible Experience

Role: heading
Name: Account settings
Description: –

Role: button
Name: Save changes
Description: –

Role: textbox
Name: Email address
Description: –

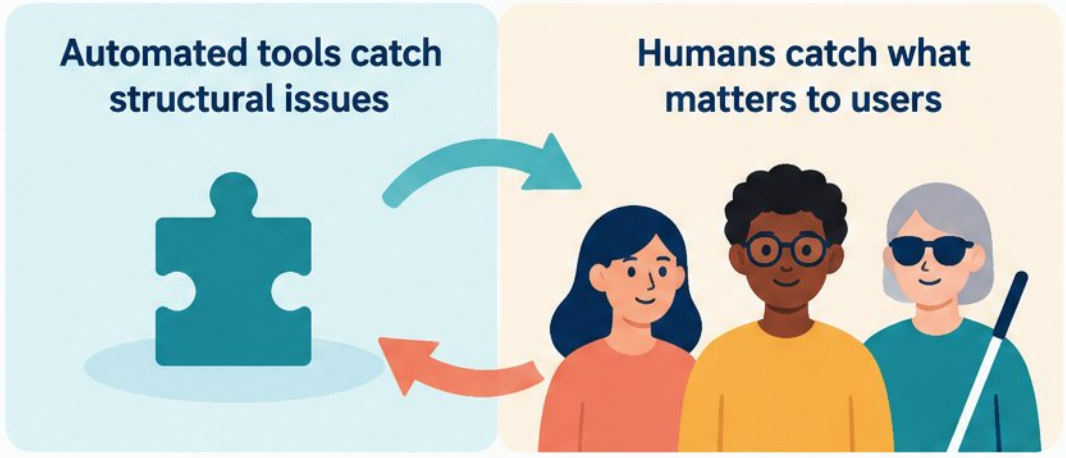
Live region update
"Settings saved" announced



- Native semantics first: real buttons, logical headings, landmarks, labeled form controls
- Accessible names must match visible labels so voice control works
- First rule of ARIA: no ARIA when native HTML does the job
- Live regions for async updates only; manage focus on SPA route changes
- Plain language, descriptive link text, captions, transcripts, accessible documents

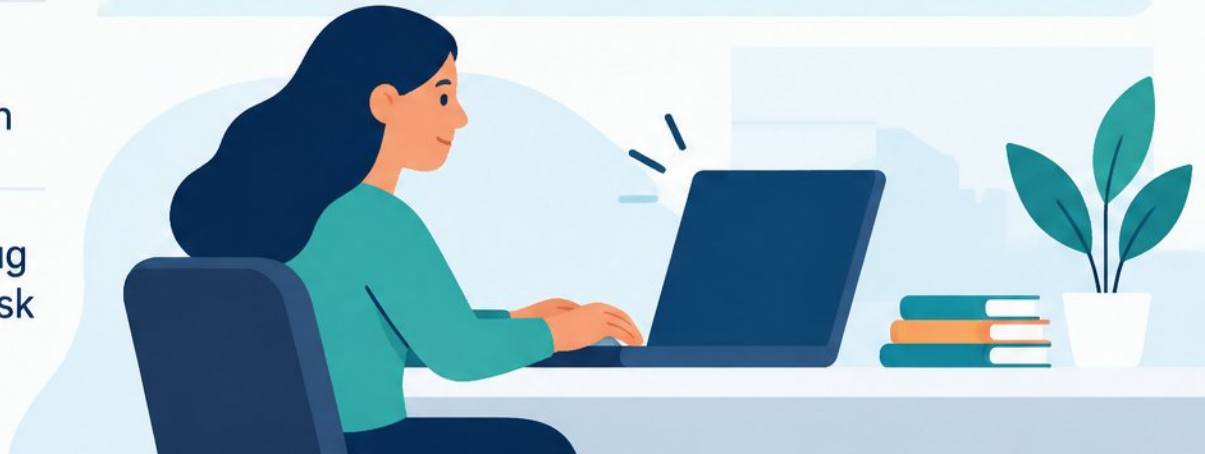
Testing Accessibility:
What Tools Catch and What Humans Must

- Automated tools (axe, Lighthouse, CI scanners) catch structural issues and prevent regressions
- Scanners can't judge meaning: alt text quality, link context, reading order, error clarity
- Manual smoke tests: keyboard-only pass, visible focus, 400% zoom, short screen reader pass
- AT coverage: NVDA/JAWS on Windows, VoiceOver on macOS/iOS, TalkBack on Android
- Test the flow, not just the page; include disabled users in research
- Prioritize by user impact; write bug reports naming the barrier and task



Manual testing in practice





Making It Stick: Roles, Process, and the Definition of Done



Product managers: own accessibility, embed it in roadmaps and sprint ceremonies

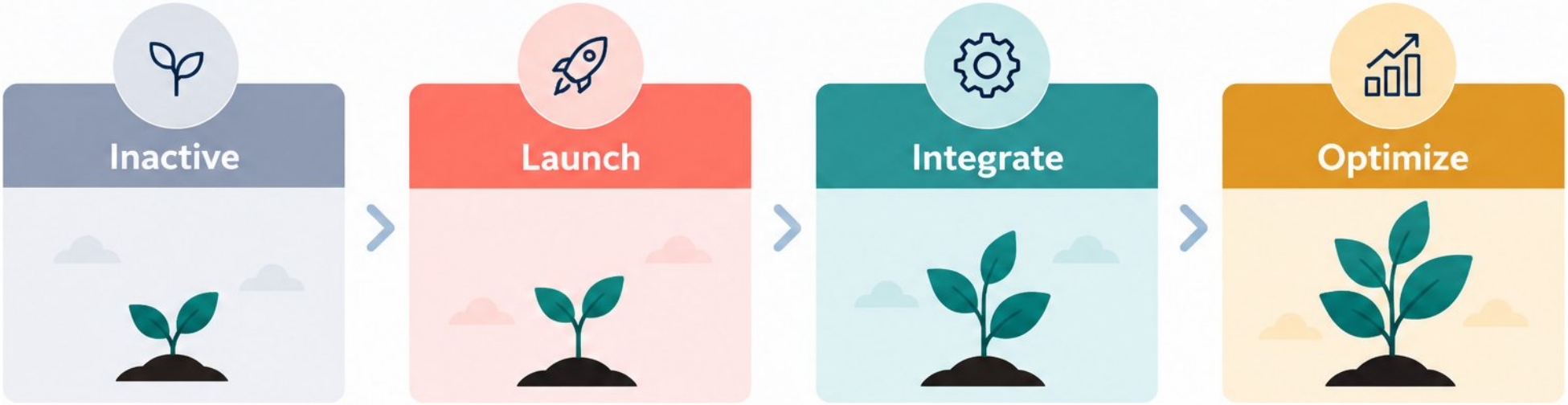


Designers and authors: annotate decisions, write plain language, add alt text and captions



Engineers and QA: use tested components, avoid unnecessary ARIA, automate and manually test

Maturity grows in stages: inactive, launch, integrate, optimize across skills, process, culture



Measure the process:
acceptance criteria in stories, keyboard and screen reader demos

Your First 30 Days: A Starter Checklist

Weeks 1–2



add accessibility acceptance criteria to the definition of done; run a keyboard-plus-screen-reader pass on your top journey

Weeks 2–3



check current work against WCAG 2.2 AA; note gaps against WCAG 2.1 assumptions still in your process

Weeks 3–4



turn on automated checks in CI; schedule manual and disabled-user testing where automation stops

Ongoing



Recruit accessibility champions per discipline; set one measurable goal for next quarter



Keep trusted, current resources shared:

W3C WAI, WCAG 2.2 Quick Reference, ARIA Authoring Practices, EN 301 549, AccessibleEU

More inclusive products.
A more equitable world.

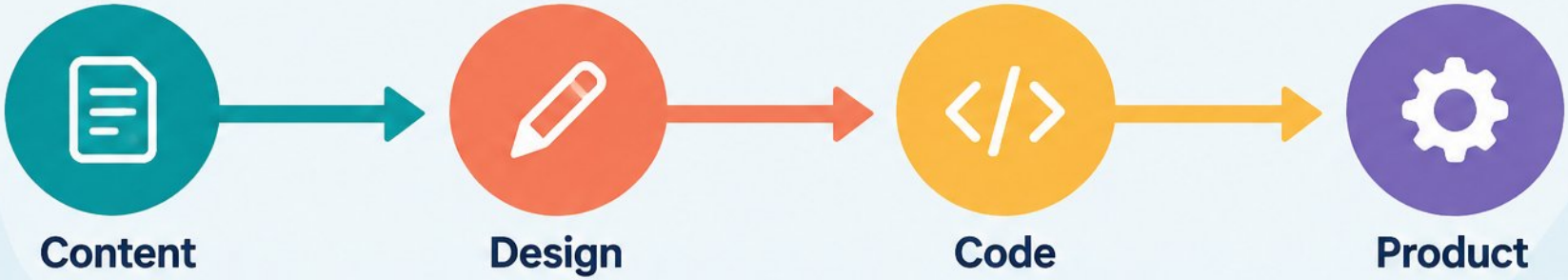
Key Takeaways and Discussion

Accessibility is a pipeline: content, design, code, and product each control one link

Guidelines work only when read through people: POUR and WCAG 2.2 map to user tasks

Regulation is converging on WCAG 2.2 AA; EN 301 549 v4.1.1 citation is expected soon

Automation and human judgment are complementary, not evidence of accessibility alone



Discussion: pick one screen, trace it through the pipeline — where does it break, and who owns the fix?





PersonWise.

PersonWise • AI digital-human course platform



Scan the code or click anywhere to watch this course live, with voice Q&A

personwise.ai/t/8ea714c7/public