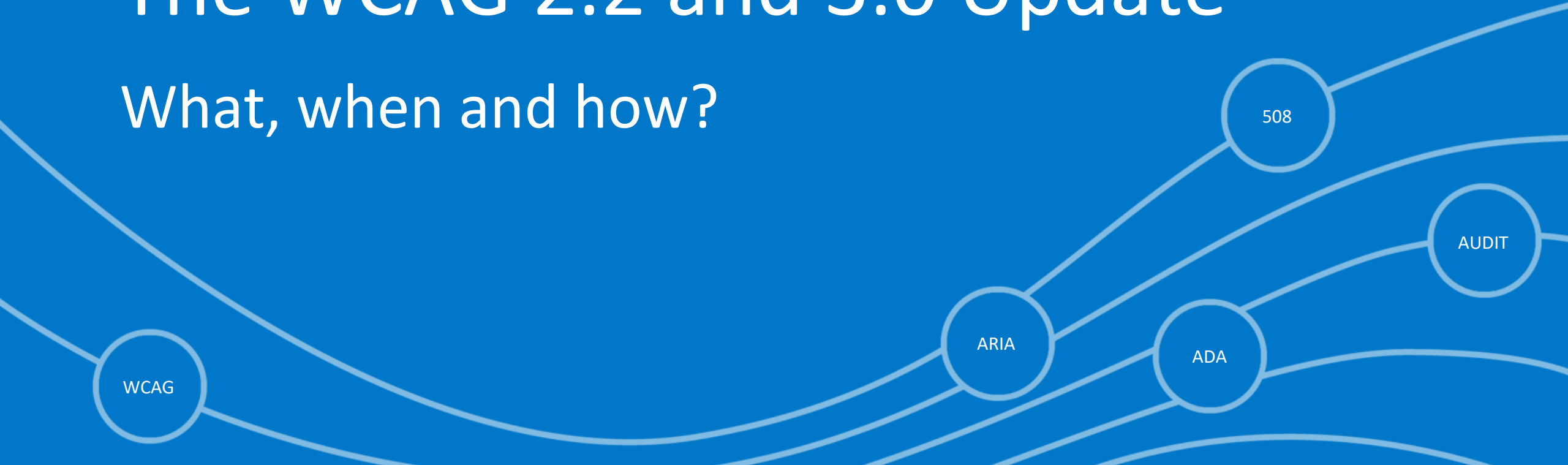
deque

The WCAG 2.2 and 3.0 Update

What, when and how?



Wilco Fiers (he/him)

- Axe-core / axe Linter product owner, Deque Europe
- Project Manager WCAG 3, W3C
- Facilitator ACT Task Force, W3C

The WCAG 2.2 and 3.0 Update

- A brief history of WCAG
- WCAG 2.2
  - When should you use it?
  - What are the new criteria
  - How do we test for them
  - Question time
- WCAG 3.0
  - New structure
  - Speculation
  - Questions & wrap-up

A Brief History of…

The Web Content Accessibility Guidelines

A Timeline of WCAG

- 1997 - 1999: WCAG 1.0
- 2001 - 2008: WCAG 2.0
- 2017 - 2018: WCAG 2.1
- 2020 - 2021: WCAG 2.2

WCAG 3.0

- 2016: Research begins
- 2021: WCAG 3.0 First Draft

Notable Adoption

- 2010: Equality Act (United Kingdom)
- 2012: ISO/IEC 40500
- 2013: Air Carrier Access Act (USA)
- 2016: Web Accessibility Directive (European Union)
- 2019: European Accessibility Act (European Union)

A more complete list can be found on w3.org/WAI/policies/

What’s New In WCAG 2.2

Strap in! It’s getting complicated.

New Success Criteria

Guideline 2.4 Navigable

2.4.11 Focus Appearance (Minimum) (Level AA)

2.4.12 Focus Appearance (Enhanced) (Level AAA)

2.4.13 Page Break Navigation (Level A)

Guideline 2.5 Input Modalities

2.5.7 Dragging Movements (Level AA)

2.5.8 Target Size (Minimum) (Level AA)

New Success Criteria

Guideline 3.2 Predictable

3.2.6 Consistent Help (Level A)

3.2.7 Visible Controls (Level AA)

Guideline 3.3 Input Assistance

3.3.7 Accessible Authentication (Level AA)

3.3.8 Accessible Authentication (No Exception) (Level AAA)

3.3.9 Redundant Entry (Level A)

2.4.11 Focus Appearance (Minimum)

When user interface components receive keyboard focus, an area of the focus indicator meets the following:

- ***Minimum area**: The area is either:*
  - *at least as large as the area of a 1 CSS pixel thick perimeter, or*
  - *at least as large as the area of a 4 CSS pixel thick line along the shortest side (…)*
- ***Contrast**: The area has a contrast ratio of at least 3:1*
- ***Adjacent contrast**: (…) a contrast ratio of at least 3:1 against the component or a thickness of at least 2 CSS pixels.*

Additionally, the item is not entirely hidden by author-created content.

2.4.11 Focus Appearance (Minimum)

Who does this help

Sighted people with mobility impairments using a keyboard-like device

- Keyboards
- switch devices
- voice input

People with low-vision using keyboard

2.4.11 Focus Appearance: Minimum Area

- At least as large as the area of a 1 CSS pixel thick perimeter

- At least as large as the area of a 4 CSS pixel thick line along the shortest side of a minimum bounding box.
- No part of the area is thinner than 2 CSS pixels

2.4.11 Focus Appearance: Contrast

The area has a **contrast ratio of at least 3:1** between the colors when the component is focused and it is not focused.

Area with **insufficient** contrast

Example Example Example

Area with **sufficient** contrast

Example Example Example

Focus indicators may include fade, provided a **minimum area** has a 3:1 contrast.

2.4.11 Focus Appearance: Adjacent Contrast

- The contrasting area also has a contrast ratio of least 3:1 against adjacent colors in the focused component

- The contrasting area has a thickness of at least 2 CSS pixels

Area with **insufficient** thickness

Area with **sufficient** thickness

2.4.11 Focus Appearance: Not Obscured

The item with focus is not entirely hidden by author-created content

In other words, when an item is focused, you should not have to scroll the page to see where it is. This is a common problem with sticky headers and footers.

2.4.11 Focus Appearance - How To Test

Step 1: From the top of the page, navigate to each element using tab, note all elements that are:

1. Fail any element that is fully obscured, or has a low contrast focus indicator
2. Note any elements with small focus indicators, or questionable contrast

Step 2: From the last focusable component on the page, use shift+tab to navigate to the top (in reverse order), note all elements that are obscured

2.4.11 Focus Appearance - How To Test

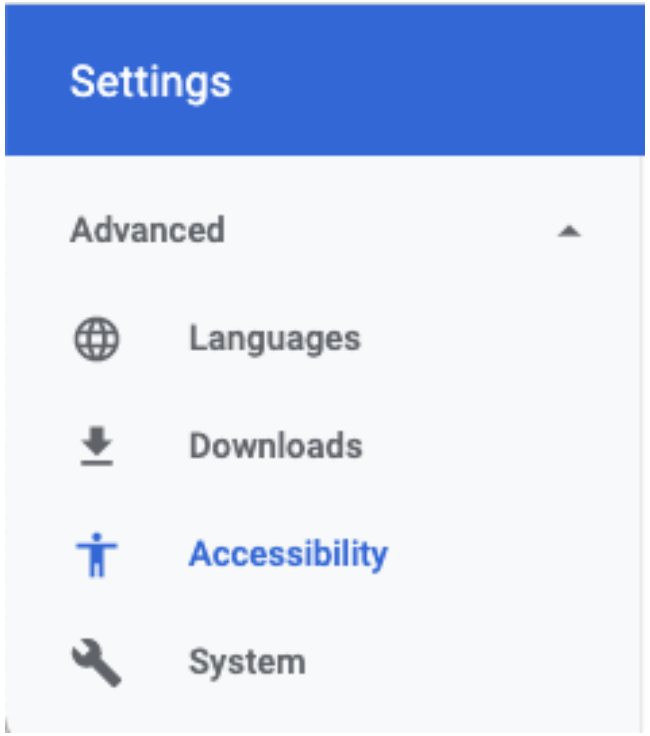
Step 3: Of the notable elements,

- Get the size using the browser's "inspect" developer tool.
- Get the contrast using a color picker and contrast analyzer
- Use screenshots to work out the exact area of the focus indicator

Good news: Coming soon, Axe DevTools Pro using the Keyboard tool will be able to test Focus Appearance **completely automatically**.

2.4.12 Focus Appearance - Browser Support

In Chrome and Edge, under Settings > Advanced > Accessibility



Show a quick highlight on the focused object

- *Fades, which is not permitted under AAA*
- *Not available in Firefox, Safari, or Chrome for Mobile*

2.4.12 Focus Appearance (Enhanced)

Difference from 2.4.11 (AA):

- **Contrast** difference has to be 4.5:1 instead of 3:1
- **Minimum area** is always as large as a 1px perimeter
- **Persistence** of the focus indicator is required

Focus indicator that is sufficient for 2.4.11 but not for 2.4.11, because its area is less than a 1 CSS pixel perimeter.

Don’t worry, it gets easier from here.

2.4.13 Page Break Navigation

For web content with page break locators, a mechanism is available to navigate to each locator.

- programmatically determinable destination markers that are arranged in a meaningful sequence to represent a locator serving the same purpose as page breaks in a printed document.

I.e. Elements with `role="doc-pagebreak"`

2.4.13 Page Break Navigation

Who does this help

People using e-books as alternatives for physical books, such as students in a classroom using a tablet for screen magnification.

How to test

Mostly only relevant when testing EPUB documents. Find a list of pages, either in the user agent or in the document itself.

2.5.7 Dragging Movements

All functionality that uses a dragging movement for operation can be achieved by a single pointer without dragging, unless dragging is essential.

Examples

- Moving through maps with on-screen arrow buttons
- Buttons in a kanban board to move cards to different columns

2.5.7 Dragging Movements

Who does this help

Holding down on a touch screen or pointer device may be difficult for some people with a mobility impairment.

How to test

Find any drag-and-drop mechanisms on the page. Check that the same can be achieved with mouse clicks (without holding down a button).

2.5.8 Target Size (Minimum)

The size of the target for pointer inputs is at least 24 by 24 CSS pixels, except where:

- **Spacing**: The target offset is at least 24 CSS pixels to every adjacent target;
- **Inline**: The target is in a sentence or block of text;
- **Necessary**: A particular presentation of the target is essential or is legally required for the information being conveyed;

2.5.5 Target Size is adjusted with “(Enhanced)”

2.5.8 Target Size (Minimum)

Who does this help

Precisely hitting a small area of a screen, either with touch or an pointer device may be difficult for some people with a mobility impairment.

Unlike *2.5.5 Target Size (Enhanced)*, this criterion if focused on not activating the wrong control, rather than activating the right control.

2.5.8 Target Size: Spacing

How to test

- Use axe-core 4.5; or
- Look at size and margin using “inspect” in the browser
- If all else fails, measure a screenshot of the browser at 100% zoom.

3.2.6 Consistent Help

If a web page contains any of the following help mechanisms, and those mechanisms are repeated on multiple web pages within a set of web pages, they occur in the same relative order to other page content, unless a change is initiated by the user:

- Human contact details;
- Human contact mechanism;
- Self-help option;
- A fully automated contact mechanism.

3.2.6 Consistent Help - How to test

Who does this help

Task completion can often be more difficult for people with disabilities. Ensuring available help is easy to find is particularly important to them.

How to test

Step 1: Find all help mechanisms, including links to help pages, contact details, chat functions, etc.

Step 2: Check that other pages that include that same help function, have it in same place on the page, relative to other repeating content.

3.2.7 Visible Controls

When user interface components are invisible until hover or focus makes them visible, provide a visible indicator that the components are available, except when:

- The same functionality is available through a component on the same page, or on a different step in a multi-step process, without requiring pointer hover or keyboard focus to make it visible;
- The component provides keyboard-only functionality;
- A mechanism is available to make the components persistently visible;
- Hiding the visual indicator is essential.

3.2.7 Visible Controls - How to test

Who does this help

People with memory issues may not be able to learn where hidden controls are.

How to test

1. Find any controls, hidden by default that are revealed on hover or focus
2. Look for any visual indication that that control is available
3. Check if any of the exceptions apply

3.3.7 Accessible Authentication

For each step in an authentication process that relies on a cognitive function test, at least one other authentication method is available that does not rely on a cognitive function test, or a mechanism is available to assist the user in completing the cognitive function test.

Exception: When the cognitive function test is to recognize common objects or content the user provided to the website.

3.3.7 Accessible Authentication

Who does this help

People with a variety cognitive abilities.

How to test

Find any authentication method that can not be used without a cognitive test.
See if the exception applies, or if a help mechanism is available.

3.3.8 Accessible Authentication (No Exceptions)

As 3.3.7, but without exception

3.3.9 Redundant Entry

Information previously entered by or provided to the user that is required to be entered again in the same process is either:

- auto-populated, or
- available for the user to select.

Except when:

- re-entering the information is essential,
- the information is required to ensure the security of the content, or
- previously entered information is no longer valid.

3.3.9 Redundant Entry

Who does this help

People who experience difficulty forming new memories may experience additional strain from having to recall or look up the same information twice.

How to test

1. Find any process that asks for the same information more than once on different pages / views.
2. Check that the repeated information can be auto-filled, or that an exception applies.

Pause For Questions

WCAG Version 3.0

W3C Accessibility Guidelines

Goals Of WCAG 3.0

- Easier to learn and understand guidelines
- Broader support disability needs
- Keep up with new technologies
- Better reflect real-world accessibility

Structure Of WCAG 3.0

WCAG 2.x	**WCAG 3.0**
Principles	*Removed*
Guidelines	Guidelines
Success criteria	Outcomes
Techniques	Methods
Non-interference requirements	Critical errors
Level A, AA, AAA	Bronze, Silver, Gold

Example Outcome: Headings organize content

Organizes content into logical blocks with headings relevant to the subsequent content. This makes locating and navigating information easier and faster.

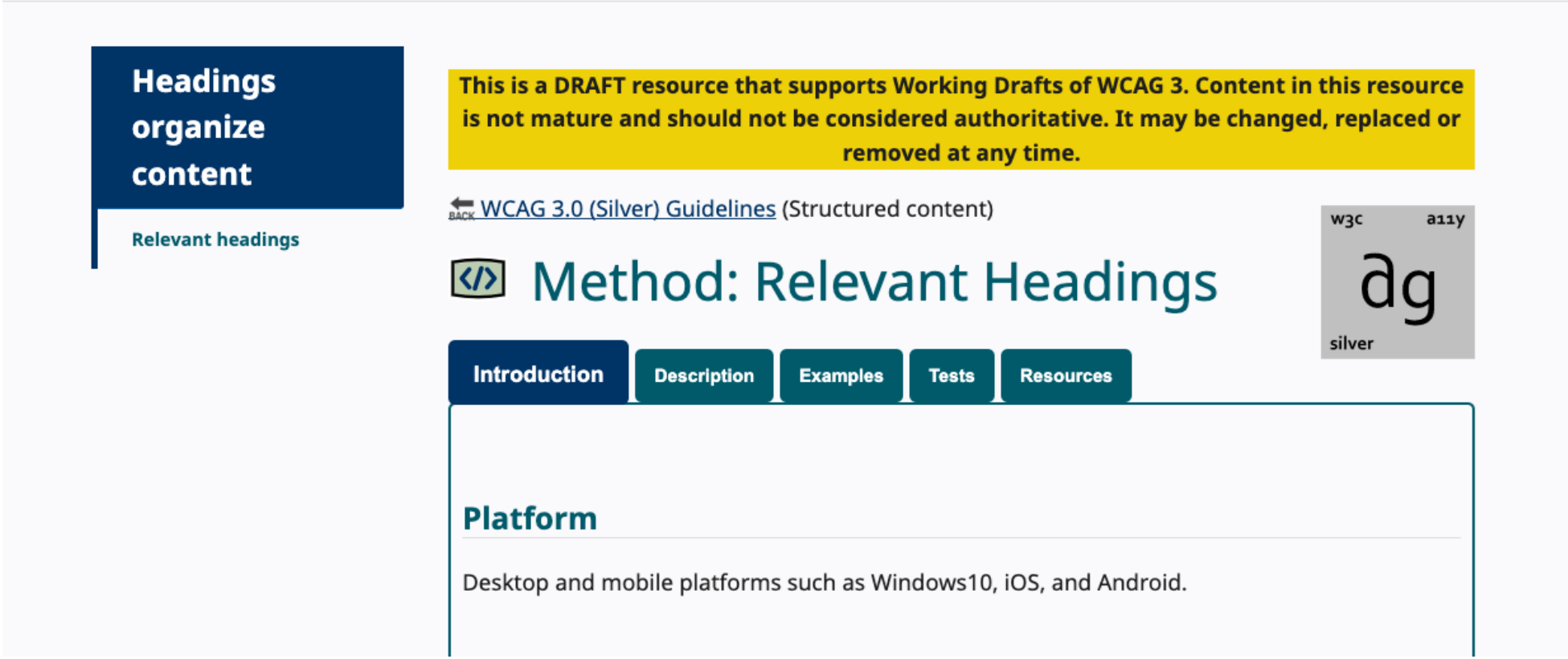
Critical errors:

One or more headings necessary to locate the content needed to complete a process are missing.

Example Outcome: Headings organize content

Rating	Criteria
Rating 0	25% or less of expected headings are present and describe the section OR there is a critical error
Rating 1	26-50% or less of expected headings
Rating 2	51-80% or less of expected headings
Rating 3	81-95% or less of expected headings
Rating 4	96-100% or less of expected headings

Example Outcome: Headings organize content

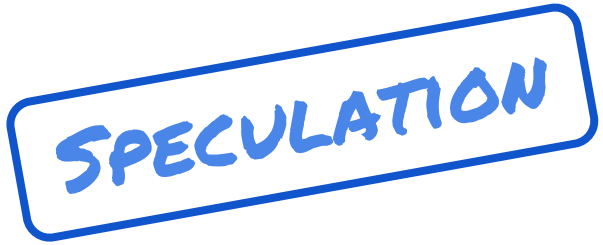


w3.org/WAI/GL/WCAG3/2020/methods/relevant-headings/

WARNING: Speculation coming up

New Features Of WCAG 3.0

- Conformance for websites and apps
- Criteria for new technologies; VR / AR
- Separate requirements for third-party content
- Sampling, scoring, metrics
- New methods of testing



WCAG 3.0 Status Levels

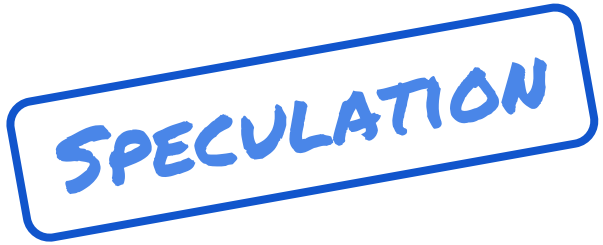
During development, WCAG 3 sections will have a status indication to show its progress

- **Placeholder**: Temporary content, hidden by default
- **Exploratory**: Very early work, hidden by default.
- **Developing**: Promising work, details still need to be worked out
- **Refining**: Largely agreed on, more public feedback needed
- **Mature**: Expected to be finalized

User Agent Methods

Browsers and assistive technologies to meet outcomes. Examples:

- Use of `lang` could become obsolete with robust language detection
- Some video could be automatically captioned
- Browser suppression of content fleshing



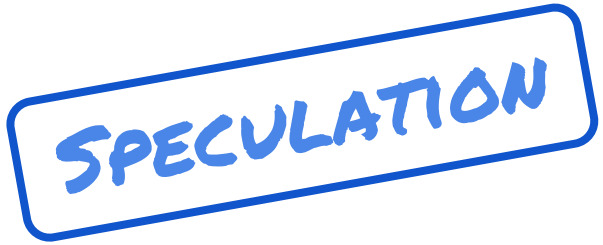
New Ways to Test: Protocols

Accessibility related guidance, not part of WCAG. Useful for:

- Guidance difficult to test consistent (e.g: task completion tests)
- Not universally applicable (e.g; accessibility needs for children)
- Niche technologies (e.g. touch screen ticket machines)

Testable through public assertions, such as:

- Metadata
- Accessibility statements



New Ways to Test: Protocols

Example of protocols:

- Plain language: plainlanguage.gov/guidelines/
- Content usable for cognitive: w3.org/TR/coga-usable/

Testable through public assertions, such as:

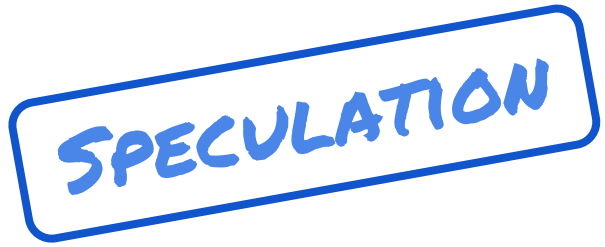
- Metadata
- Accessibility statements

The “Timeline” Problem

Assuming no major setbacks, WCAG 3.0 might be done by **2028**

- Rewriting just WCAG 2.2 takes several years
- Every new feature takes months, if not years
- There are only so many of us working on this

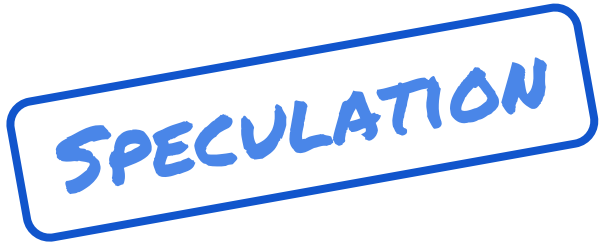
That’s a long wait; and it may be much longer than that.



Possible “Timeline” Solutions

Parts of WCAG 3.0 could be released sooner, while the whole is being worked out

- Limit WCAG 3.0, and publish the rest in a 3.1 update
- Publish a “WCAG 2.9” transition
- Publish WCAG 3.0 as modules; For example:
  - WCAG 3.0: Core
  - WCAG 3.0: Sites & Apps
  - WCAG 3.0: Media & XR



Getting Involved

Read the latest draft:

w3.org/TR/wcag-3.0/

Leave a review, or join the conversation:

github.com/w3c/silver/issues

Become a member:

w3.org/community/silver

Connect with us

- @dequesystems
- dequelabs
- deque-systems-inc
- dequesystemsinc