

# Guidance on Applying WCAG 2.2 to Non-Web Information and Communications Technologies (WCAG2ICT)



W3C Group Draft Note 15 August 2023

## ▼ More details about this document

### This version:

<https://www.w3.org/TR/2023/DNOTE-wcag2ict-20230815/>

### Latest published version:

<https://www.w3.org/TR/wcag2ict/>

### Latest editor's draft:

<https://w3c.github.io/wcag2ict/>

### History:

<https://www.w3.org/standards/history/wcag2ict/>  
[Commit history](#)

### Latest Recommendation:

<https://www.w3.org/TR/wcag2ict/>

### Editors:

[Mary Jo Mueller](#) (IBM)  
[Chris Loisel](#) (Oracle Corporation)  
[Phil Day](#) (NCR)

### Former editors:

Michael Cooper (W3C)  
Peter Korn (Amazon)  
Andi Snow-Weaver (IBM)  
Gregg Vanderheiden (Invited Expert, Trace Research and Development Center)

### Feedback:

[GitHub w3c/wcag2ict](#) ([pull requests](#), [new issue](#), [open issues](#))

[Copyright](#) © 2022-2023 [World Wide Web Consortium](#). W3C<sup>®</sup> [liability](#), [trademark](#) and [document use](#) rules apply.

---

## Abstract

This document, “Guidance on Applying WCAG 2.2 to Non-Web Information and Communications Technologies (WCAG2ICT)” describes how the Web Content Accessibility Guidelines (WCAG) 2.2 [[WCAG22](#)] and its principles, guidelines, and success criteria can be applied to non-web Information and Communications Technologies (ICT), specifically to non-web documents and software. It provides informative guidance (guidance that is not normative and does not set requirements).

This document is part of a series of technical and educational documents published by the [W3C Web Accessibility Initiative \(WAI\)](#) and available from the [WCAG2ICT Overview](#).

## Status of This Document

*This section describes the status of this document at the time of its publication. A list of current [W3C](#) publications and the latest revision of this technical report can be found in the [W3C technical reports index](#) at <https://www.w3.org/TR/>.*

This is a technical report on Applying WCAG 2.2 to Non-Web Information and Communications Technologies (WCAG2ICT). The intent of this work is to update the existing guidance based on new WCAG 2.1 and 2.2 success criteria.

The current draft includes guidance for WCAG 2.1 success criteria. Guidance on WCAG 2.2 success criteria will be added later this year as WCAG 2.2 becomes a [W3C Recommendation](#). The next draft will also address open issues on WCAG 2.0 criteria.

The group is seeking feedback on the following aspects:

- New added guidance for WCAG 2.1 success criteria and glossary term definitions
- Updates to the closed functionality guidance for new WCAG 2.1 success criteria

The group is already aware of formatting differences in how quoted passages from WCAG are included in comparison to the previous WCAG2ICT Working Group Note.

Currently there are known issues in the quoted content from WCAG and Understanding Intent sections:

- Anchor links from the included WCAG content to sections that do not exist in the WCAG2ICT document don't work
- The words "Note" and "Example" are missing from notes and examples of quoted

## WCAG passages

The group is currently working on updates to the format, styling, and linking for these passages, and expects to refine these in a future draft.

This document was published by the [Accessibility Guidelines Working Group](#) as a Group Draft Note using the [Note track](#).

Group Draft Notes are not endorsed by W3C nor its Members.

This is a draft document and may be updated, replaced or obsoleted by other documents at any time. It is inappropriate to cite this document as other than work in progress.

The [1 August 2017 W3C Patent Policy](#) does not carry any licensing requirements or commitments on this document.

This document is governed by the [12 June 2023 W3C Process Document](#).

## Table of Contents

### Abstract

### Status of This Document

- 1. Introduction**
  - 1.1 Background
  - 1.2 Guidance in this Document
  - 1.3 Excluded from Scope
  - 1.4 Document Overview
  - 1.5 Document Conventions
  - 1.6 Comparison with the 2013 WCAG2ICT Note
- 2. Key Terms**
  - 2.1 Accessibility Services of Platform Software
  - 2.2 Content (on and off the Web)
  - 2.3 Document
  - 2.4 Set of Documents
  - 2.5 Set of Software Programs
  - 2.6 Software
  - 2.7 User Agent

- 3. Closed Functionality**
- 4. Text / Command-line / Terminal Applications and Interfaces**
- 5. Comments on Conformance**
- 6. Comments by Guideline and Success Criterion**
  - 6.1 Perceivable
    - 6.1.2 Text Alternatives
      - 6.1.2.2 Non-text Content
    - 6.1.3 Time-based Media
      - 6.1.3.2 Audio-only and Video-only (Prerecorded)
      - 6.1.3.3 Captions (Prerecorded)
      - 6.1.3.4 Audio Description or Media Alternative (Prerecorded)
      - 6.1.3.5 Captions (Live)
      - 6.1.3.6 Audio Description (Prerecorded)
    - 6.1.4 Adaptable
      - 6.1.4.2 Info and Relationships
      - 6.1.4.3 Meaningful Sequence
      - 6.1.4.4 Sensory Characteristics
      - 6.1.4.5 Orientation
      - 6.1.4.6 Identify Input Purpose
    - 6.1.5 Distinguishable
      - 6.1.5.2 Use of Color
      - 6.1.5.3 Audio Control
      - 6.1.5.4 Contrast (Minimum)
      - 6.1.5.5 Resize Text
      - 6.1.5.6 Images of Text
      - 6.1.5.7 Reflow
      - 6.1.5.8 Non-text Contrast
      - 6.1.5.9 Text Spacing
      - 6.1.5.10 Content on Hover or Focus
  - 6.2 Operable
    - 6.2.2 Keyboard Accessible
      - 6.2.2.2 Keyboard
      - 6.2.2.3 No Keyboard Trap
      - 6.2.2.4 Character Key Shortcuts
    - 6.2.3 Enough Time
      - 6.2.3.2 Timing Adjustable

- 6.2.3.3 Pause, Stop, Hide
- 6.2.4 Seizures and Physical Reactions
  - 6.2.4.2 Three Flashes or Below Threshold
- 6.2.5 Navigable
  - 6.2.5.2 Bypass Blocks
  - 6.2.5.3 Page Titled
  - 6.2.5.4 Focus Order
  - 6.2.5.5 Link Purpose (In Context)
  - 6.2.5.6 Multiple Ways
  - 6.2.5.7 Headings and Labels
  - 6.2.5.8 Focus Visible
  - 6.2.5.9 Focus Not Obscured (Minimum)
- 6.2.6 Input Modalities
  - 6.2.6.2 Pointer Gestures
  - 6.2.6.3 Pointer Cancellation
  - 6.2.6.4 Label in Name
  - 6.2.6.5 Motion Actuation
  - 6.2.6.6 Dragging Movements
  - 6.2.6.7 Target Size (Minimum)
- 6.3 Understandable
  - 6.3.2 Readable
    - 6.3.2.2 Language of Page
    - 6.3.2.3 Language of Parts
  - 6.3.3 Predictable
    - 6.3.3.2 On Focus
    - 6.3.3.3 On Input
    - 6.3.3.4 Consistent Navigation
    - 6.3.3.5 Consistent Identification
    - 6.3.3.6 Consistent Help
  - 6.3.4 Input Assistance
    - 6.3.4.2 Error Identification
    - 6.3.4.3 Labels or Instructions
    - 6.3.4.4 Error Suggestion
    - 6.3.4.5 Error Prevention (Legal, Financial, Data)
    - 6.3.4.6 Redundant Entry
    - 6.3.4.7 Accessible Authentication (Minimum)
- 6.4 Robust
  - 6.4.2 Compatible
    - 6.4.2.2 Parsing (Obsolete and removed)

6.4.2.3 Name, Role, Value

6.4.2.4 Status Messages

## **7. Comments on Definitions in WCAG 2.2 Glossary**

7.1 Glossary Items that Apply to All Technologies

7.2 Glossary Items Used only in AAA Success Criteria

7.3 Glossary Items with Specific Guidance

7.3.1 accessibility supported

7.3.2 ambiguous to users in general

7.3.3 assistive technology

7.3.4 changes of context

7.3.5 conformance

7.3.6 conforming alternate version

7.3.7 content

7.3.8 contrast ratio

7.3.9 css pixel

7.3.10 down-event

7.3.11 general flash and red flash thresholds

7.3.12 input error

7.3.13 keyboard interface

7.3.14 keyboard shortcut

7.3.15 label

7.3.16 name

7.3.17 programmatically determined

7.3.18 programmatically set

7.3.19 relative luminance

7.3.20 role

7.3.21 same functionality

7.3.22 satisfies a success criterion

7.3.23 set of web pages

7.3.24 structure

7.3.25 style property

7.3.26 target

7.3.27 technology

7.3.28 up-event

7.3.29 user agent

7.3.30 user interface component

7.3.31 viewport

7.3.32 web page

## 8. Privacy Considerations

## 9. Security Considerations

### A. Success Criteria Problematic for Closed Functionality

### B. Background on Text / Command-line / Terminal Applications and Interfaces

B.1 How text interfaces are realized

B.2 How text applications have been made accessible via assistive technology

B.3 Applying WCAG 2.2 to text applications

### C. Acknowledgements

C.1 Participants of the WCAG2ICT Task Force Active in the Development of this Document

C.2 Participants in the AG Working Group that Actively Reviewed and Contributed

C.3 Participants in the APA Working Group that Contributed

C.4 Previous Contributors

C.5 Enabling Funders

### D. References

D.1 Informative references

## § 1. Introduction

### § 1.1 Background

This document is an update to a [W3C Working Group Note](#) to incorporate new guidelines, success criteria, and definitions added in WCAG 2.1 and 2.2.

#### [Guidance on Applying WCAG 2.0 to Non-Web Information and Communications](#)

[Technologies \(WCAG2ICT\)](#), approved in September 2013, described how WCAG 2.0 could be applied to non-web documents and software. WCAG2ICT was organized to mirror WCAG's sections: Perceivable, Operable, Understandable, and Robust. WCAG2ICT clarified when and how WCAG success criteria should be applied to non-web documents and software. Some

were applicable without modification and some were applicable with edits and/or notes. Glossary terms were also reviewed. Level AAA Success Criteria were not addressed in the 2013 WCAG2ICT Working Group Note.

The 2013 WCAG2ICT has been relied upon in regulations and legislation. One example is EN 301 549<sup>[^1]</sup> (Europe) and other standards that reference or incorporate EN 301 549 (e.g., India, Kenya, Australia). Another example is Section 508 (U.S.) [Application of WCAG 2.0 to Non-Web ICT](#), which looked to WCAG2ICT for detailed direction with providing specific guidance and exceptions to particular criteria from being applied to non-web technology. Section 508 incorporated by reference WCAG as the [Accessibility Standard applicable to non-web documents](#) and requires [WCAG Conformance for non-web software](#).

[^1]: EN 301 549 [V3.2.1](#) 2.2 Informative references, p. 13 [i26].

## § 1.2 Guidance in this Document

### EDITOR'S NOTE

This section contains a first pass of updates. It will be re-examined once the Task Force has finished analyzing all of the new WCAG criteria to ensure the summarization of applicability of WCAG criteria to non-web documents and software is accurate.

This document provides informative guidance (guidance that is not [normative](#) and that does not set requirements) with regard to the interpretation and application of [Web Content Accessibility Guidelines \(WCAG\) 2.2](#) [WCAG22] to non-web information and communications technologies (ICT). This document is a [Working Group Note](#) (in contrast to WCAG 2.1 and WCAG 2.2, which are [W3C Recommendations](#)). Specifically, this document provides informative guidance on applying WCAG 2.2 Level A and AA success criteria to non-web ICT, specifically to non-web documents and software.

This document is intended to help clarify how to use WCAG 2.2 to make non-web documents and software more accessible to people with disabilities. Addressing accessibility involves addressing the needs of people with auditory, cognitive, neurological, physical, speech, and visual disabilities, and the needs of people with accessibility requirements due to the effects of aging. Although this document covers a wide range of issues, it is not able to address all the needs of all people with disabilities. Because WCAG 2.2 was developed for the Web, addressing accessibility for non-web documents and software may involve requirements and considerations beyond those

included in this document. Authors and developers are encouraged to seek relevant advice about current best practices to ensure that non-web documents and software are accessible, as much as possible, to people with disabilities.

While WCAG 2.2 was designed to be technology-neutral, it assumes the presence of a “user agent” such as a browser, media player, or assistive technology as a means to access web content. Therefore, the application of WCAG 2.2 to documents and software in non-web contexts required some interpretation in order to determine how the intent of each WCAG 2.2 success criterion could be met in these different contexts of use. Therefore, the bulk of the Task Force's work involved evaluating how each WCAG 2.2 success criterion would apply in the context of non-web ICT, if it were applied to non-web ICT.

The Task Force found that the majority of success criteria from WCAG 2.2 can apply to non-web documents and software with either no or minimal changes. Since many of the Level A and AA success criteria do not include any web related terms, they apply directly as written and as described in the “Intent” sections from the [Understanding WCAG 2.2 \[UNDERSTANDING-WCAG22\]](#) resource. Additional notes were provided, as needed, to provide assistance in applying them to non-web documents and software.

When certain Web-specific terms or phrases like “web page(s)” were used in success criteria, those were replaced with non-web terms or phrases like “non-web document(s) and software”. Additional notes were also provided to explain the terminology replacements.

A small number of success criteria are written to apply to “a set of web pages” or “multiple web pages” and require all pages in the set to share some characteristic or behavior. Since the unit of conformance in WCAG 2.2 is a single web page, the task force agreed that the equivalent unit of conformance for non-web documents is a single document. It follows that an equivalent unit of evaluation for a “set of web pages” would be a “set of documents”. Since it isn't possible to unambiguously carve up non-web software into discrete pieces, a single “web page” was equated to a “software program” and a “set of web pages” was equated to a “set of software programs. Both of these new terms are defined in the Key Terms section of this document. See “[set of documents](#)” and “[set of software programs](#)” to determine when a group of documents or pieces of software are considered a set.

#### NOTE

Sets of software that meet this definition appear to be extremely rare.

The glossary terms were also reviewed and most of them applied to non-Web documents

and software, as written. Some applied with additional notes or edits (largely related to phrases like “Web page(s)”), and a small number of terms were only used in Level AAA success criteria which are not addressed by the WCAG2ICT Note at this time.

## § 1.3 Excluded from Scope

The following are out of scope for this document:

- This document does not seek to determine which WCAG 2.2 provisions (principles, guidelines, or success criteria) should or should not apply to non-web documents and software, but rather how they would apply, if applied.
- This document does not propose changes to WCAG 2.2 or its supporting documents; it does not include interpretations for implementing WCAG 2.2 in web technologies. During the development of this document, the WCAG2ICT Task Force did seek clarification on the intent of a number of the success criteria, which led to clarifications in the Understanding WCAG 2.2 document.
- This document is not sufficient by itself to ensure accessibility in non-web documents and software. As a web standard, WCAG does not fully cover all accessibility requirements for non-user interface aspects of platforms, user-interface components as individual items, nor closed product software (where there is no Assistive Technology to communicate programmatic information).
- This document does not comment on hardware aspects of products, because the basic constructs on which WCAG 2.2 is built do not apply to these.
- This document does not provide supporting techniques for implementing WCAG 2.2 in non-web documents and software.
- This document is purely an informative Note about non-web ICT, not a standard, so it does not describe how non-web ICT should conform to it.

## § 1.4 Document Overview

This document includes text quoted from the WCAG 2.2 principles, guidelines, and success criteria, without any changes. It also includes excerpted text from the “Intent” sections of the WCAG 2.2 supporting document [Understanding WCAG 2.2 \(Public Review Draft\)](#)

[[UNDERSTANDING-WCAG22](#)]. The guidance provided by this document for each success criterion is preceded by a heading beginning with “Additional Guidance...”. This guidance was created by the WCAG2ICT Task Force, then reviewed and approved by the AG Working Group.

Additional supporting documents for WCAG 2.2, such as the [WCAG 2 Overview](#), [Techniques for WCAG 2.2](#) [WCAG22-TECHS], and [How to Meet WCAG \(Quick Reference\)](#), remain available for web content, but have not been changed to apply to non-web documents and software.

## § 1.5 Document Conventions

### EDITOR'S NOTE

The visual styling and programmatic structure details for calling out content in this section are current for this draft. This section will be revisited when further style details are worked out.

The following stylistic conventions are used in this document:

- Quotes from WCAG 2.2 and Understanding WCAG 2.2 are in `<blockquote>` elements and visually styled with a gray bar on the left, and immediately follow the heading for the principle, guideline, or success criterion.
- Additional guidance provided by this document begins with the phrase “Guidance When Applying” and has no special visual styling.
- Replacement text that is presented to show how an SC would read as modified by the advice in this document are in `<ins>` elements visually styled as bold green text with a dotted underline.
- Notes are slightly inset and begin with the phrase “NOTE”. Each note is in its own inset box styled in pale green with a darker green line on the left side of the box.
- References to glossary items from WCAG 2.2 are presented in `<cite>` elements visually styled as ordinary text with a dotted underline, and contain title attributes noting these are WCAG definitions. They turn blue with a yellow background when mouse or keyboard focus is placed over them.
- References to glossary items in this document are presented in `<cite>` elements visually styled as ordinary text with a dark gray underline.

- Hereafter, the short title “WCAG2ICT” is used to reference this document.

## § 1.6 Comparison with the 2013 WCAG2ICT Note

### EDITOR'S NOTE

The WCAG2CIT Task Force has incorporated all of the new WCAG 2.1 guidelines, criteria and glossary terms. The next draft version will incorporate new WCAG 2.2 criteria and glossary terms as well as address open issues on any of the content in the document.

The following changes and additions have been made to update the 2013 WCAG2ICT document:

- New [Background](#) section to explain the history and known uses of WCAG2ICT
- New WCAG 2.1 Success Criteria and Guidelines
  - [Success Criterion 1.3.4 Orientation](#)
  - [Success Criterion 1.3.5 Identify Input Purpose](#)
  - [Success Criterion 1.4.10 Reflow](#)
  - [Success Criterion 1.4.11 Non-text Contrast](#)
  - [Success Criterion 1.4.12 Text Spacing](#)
  - [Success Criterion 1.4.13 Content on Hover or Focus](#)
  - [Success Criterion 2.1.4 Character Key Shortcuts](#)
  - [Guideline 2.5 Input Modalities](#)
  - [Success Criterion 2.5.1 Pointer Gestures](#)
  - [Success Criterion 2.5.2 Pointer Cancellation](#)
  - [Success Criterion 2.5.3 Label in Name](#)
  - [Success Criterion 2.5.4 Motion Actuation](#)
  - [Success Criterion 4.1.3 Status Messages](#)
- New WCAG 2.2 Success Criteria
  - [Success Criterion 2.5.8 Target Size \(Minimum\)](#)
- New terms

- pointer input, process, single pointer, state, status message were added to [Glossary Items that Apply to All Technologies](#)
- motion animation, region, and user inactivity were added to [Glossary Items Used only in AAA Success Criteria](#)
- [css pixel](#)
- [down event](#)
- [keyboard shortcut](#)
- [style property](#)
- [target](#)
- [up event](#)
- Updated terms
  - [set of web pages](#)
  - [set of non-web documents](#)
  - [set of software programs](#)
- Updated sections

#### NOTE

In this draft, most of the existing sections have undergone WCAG2ICT Task Force review and updates. Many sections required only minor editorial and link URL updates, such as the guidance for each WCAG 2.0 success criteria. Any sections that have not been fully updated have editor's notes to reflect their current status.

## § 2. Key Terms

There are two key glossary terms from WCAG 2.2 that need to be interpreted significantly differently when applied to non-web ICT. These are: “content” and “user agent”. In addition, the glossary term “Web page” in WCAG 2.2 is replaced with newly defined terms “document” and “software”, and both “set of web pages” and “multiple web pages” are replaced with the newly defined terms “set of documents” and “set of software programs”. Finally, since non-Web software doesn't leverage the WCAG 2.2 notion of a user agent, we introduced the new term “accessibility services of platform software”. The remaining glossary terms from WCAG 2.2 are addressed in [Chapter 7 Comments on Definitions in](#)

[WCAG 2.2 Glossary](#). Terms defined and used in WCAG2ICT are applicable only to the interpretation of the guidance in this document. The particular definitions should not be interpreted as having applicability to situations beyond the scope of WCAG2ICT. Further information on usage of these terms follows.

## § 2.1 Accessibility Services of Platform Software

The term **accessibility services of platform software**, as used in WCAG2ICT, has the meaning below:

### **accessibility services of platform software (as used in WCAG2ICT)**

services provided by an operating system, [user agent](#), or other platform software that enable non-web [documents](#) or [software](#) to expose information about the user interface and events to assistive technologies and accessibility features of software

#### NOTE

These services are commonly provided in the form of accessibility APIs (application programming interfaces), and they provide two-way communication with assistive technologies, including exposing information about objects and events.

## § 2.2 Content (on and off the Web)

WCAG 2.2 defines **content** as:

information and sensory experience to be communicated to the user by means of a [user agent](#), including code or markup that defines the content's [structure](#), [presentation](#), and interactions

For non-web content it is necessary to view this a bit more broadly. Within WCAG2ICT, the term “content” is used as follows:

### **content (non-web content) (as used in WCAG2ICT)**

information and sensory experience to be communicated to the user by means of [\[software\]](#), including code or markup that defines the content's [structure](#),

[presentation](#), and interactions

#### NOTE

Non-web content occurs in two places; documents and software. When content occurs in a document, a user agent is needed in order to communicate the content's information and sensory experience to the user. When content occurs in software, a separate user agent isn't required—the software itself performs that function.

Within WCAG2ICT wherever “content” or “web content” appears in a success criterion it is replaced with “content” using the definition above.

## § 2.3 Document

The term **document**, as used in WCAG2ICT, has the meaning below:

#### **document (as used in WCAG2ICT)**

assembly of [content](#), such as a file, set of files, or streamed media that functions as a single item rather than a collection, that is not part of software and that does not include its own user agent

#### NOTE 1

A document always requires a user agent to present its content to the user.

#### NOTE 2

Letters, spreadsheets, emails, books, pictures, presentations, and movies are examples of documents.

### NOTE 3

Software configuration and storage files such as databases and virus definitions, as well as computer instruction files such as source code, batch/script files, and firmware, are examples of files that function as part of [software](#) and thus are not examples of documents. If and where software retrieves “information and sensory experience to be communicated to the user” from such files, it is just another part of the content that occurs in software and is covered by WCAG2ICT like any other parts of the software. Where such files contain one or more embedded documents, the embedded documents remain documents under this definition.

### NOTE 4

A collection of files zipped together into an archive, stored within a single virtual hard drive file, or stored in a single "encrypted file system" file, do not constitute a single document. The software that archives/encrypts those files or manages the contents of the virtual hard drive does not function as a user agent for the individually collected files in that collection because that software is not providing a non-fully functioning presentation of that content.

### NOTE 5

Anything that can present its own content without involving a user agent, such as a self playing book, is not a document but is software.

### NOTE 6

A single document may be composed of multiple files such as the video content, closed caption text, etc. This fact is not usually apparent to the end-user consuming the document / content. This is similar to how a single web page can be composed of content from multiple URIs (e.g. the page text, images, the JavaScript, a CSS file etc.).

Example: An assembly of files that represented the video, audio, captions and timing files for a movie would be a document.

Counterexample: A binder file used to bind together the various exhibits for a legal case would not be a document.

## § 2.4 Set of Documents

The term **set of documents**, as used in WCAG2ICT, has the meaning below:

### **set of documents (non-web) (as used in WCAG2ICT)**

collection of documents that share a common purpose, are created by the same author, group or organization [and that are published together, and the documents all refer to each other by name or link]

#### NOTE 1

Republishing or bundling previously published documents as a collection does not constitute a set of documents.

#### NOTE 2

If a set is broken apart, the individual parts are no longer part of a set, and would be evaluated as any other individual document is evaluated.

Example: One example of a set of documents would be a three-part report where each part is a separate file. The table of contents is repeated at the beginning of each file to enable navigation to the other parts.

## § 2.5 Set of Software Programs

The term **set of software programs**, as used in WCAG2ICT, has the meaning below:

**set of software programs (as used in WCAG2ICT)**

collection of **software programs** that share a common purpose, are created by the same author, group or organization **[and that are distributed together and can be launched and used independently from each other, but are interlinked each with every other one such that users can navigate from one program to another via a consistent method that appears in each member of the set]**

**NOTE 1**

Although "sets of web pages" occur frequently, "sets of software programs" appear to be extremely rare.

**NOTE 2**

Redistributing or bundling previously distributed software as a collection does not constitute a set of software programs.

**NOTE 3**

Consistent does not mean identical. For example, if a list of choices is provided it might not include the name of the current program.

**NOTE 4**

If a member of the set is separated from the set, it is no longer part of a set, and would be evaluated as any other individual software program.

**NOTE 5**

Any software program that is not part of a set, per this definition, would automatically satisfy any success criterion that is specified to apply to "sets of" software (as is true for any success criterion that is scoped to only apply to some other type of content).

#### NOTE 6

If there is any ambiguity whether the group is a set, then the group is not a set.

#### NOTE 7

If there is no independent method to launch the software programs (as is common in closed products), those programs would not meet the definition of a "set of software programs".

#### NOTE 8

Although the term "software" is used throughout this document because this would apply to stand alone software programs as well as individual software components and the software components in software-hardware combinations, the concept of "set of software programs" would only apply (by definition) to programs that can be launched separately from each other. Therefore, for the provisions that use the phrase "set of" (success criteria 2.4.1, 2.4.5, 3.2.3, and 3.2.4), the phrase "set of software programs" is used.

Example: One example of a set of software programs would be a group of programs that can be launched and used separately but are distributed together and all have a menu that allows users to launch, or switch to, each of the other programs in the group.

Counterexamples: Examples of things that are **not** sets of software programs:

- A suite of programs for authoring different types of documents (text, spreadsheets, presentations, etc.) where the programs don't provide an explicit, consistent means to launch, or switch to, each of the other programs in the group.
- An office package consisting of multiple programs that launches as a single program that provides multiple functionalities such as writing, spreadsheet, etc., but the only way to navigate between programs is to open a document in one of the programs.
- A bundle of software programs that is sold together but the only way to navigate between the programs in the bundle is to use a platform software level menu to navigate between them (and not via a menu provided by each program that allows you to navigate to just the other programs in this bundle).
- A group of programs that was a set, but the programs have been moved to separate locations so that their “set” behaviors were disrupted and no longer work. Even though they *were* a set at one time, because they are no longer installed as a set they no longer *are* a set and would not need to meet any success criteria that apply to sets of software.

## § 2.6 Software

The term **software** as used in WCAG2ICT, has the meaning below:

### **software (as used in WCAG2ICT)**

software products or software aspects of hardware-software products that have a user interface and do not require a separate [user agent](#) to present any of its [content](#)

#### NOTE 1

For software, the user interface and any other embedded content is covered by these guidelines. The software provides a function equivalent to a user agent for the embedded content.

#### NOTE 2

Software without a user interface does not have content and is not covered by these guidelines. For example, driver software with no user interface would not be covered.

#### NOTE 3

Because software with a user interface provides a function equivalent to a user agent in addition to content, the application of some WCAG 2.2 success criteria would be different for content embedded in software versus content in a document, where it is viewed through a separate user agent (e.g. browser, player, viewer, etc.).

## § 2.7 User Agent

WCAG 2.2 defines **user agent** as:

#### **user agent**

any software that retrieves and presents Web content for users

Example: Web browsers, media players, plug-ins, and other programs—including [assistive technologies](#)—that help in retrieving, rendering, and interacting with Web content.

For non-web ICT, “user agent” needs to be viewed differently. In WCAG 2.2, the term “user agent” only refers to retrieval and display of web content. For non-web ICT, the term “user agent” refers to retrieval and display of separate content that is *not on the Web*, which

WCAG2ICT refers to as a “document”. Within WCAG2ICT, the term “user agent” is used as follows:

**user agent (as used in WCAG2ICT)**

any [software](#) that retrieves and presents **[documents]** for users

**NOTE 1**

Software that only displays the [content](#) contained within it is not considered to be a user agent. It is just considered to be software.

**NOTE 2**

An example of software that is not a user agent is a calculator application that doesn't retrieve the calculations from outside the software to present it to a user. In this case, the calculator software is not a user agent, it is simply software with a user interface.

**NOTE 3**

Software that only shows a preview of content such as a thumbnail or other non-fully functioning presentation is not providing user agent functionality.

## § 3. Closed Functionality

As noted in the Introduction, WCAG 2.2 assumes the presence of a “user agent” such as a browser, media player, or assistive technology as a means to access web content. Furthermore, many of the success criteria in WCAG 2.2 assume web content will be accessed by ICT that has assistive technologies connected to it, where the assistive technologies present the web content to the people with disabilities in accessible form. ICT products with “closed functionality” do not allow the use of some assistive technologies for all of their functions. In many cases such ICT products also lack a “user agent” or their equivalent. As a result, ICT following these success criteria by themselves will not make information accessible on ICT with closed functionality. Something else needs to be provided or be required in order to make the information addressed in these success criteria accessible. It is outside the [WCAG2ICT Task Force Work Statement](#) to say

what the additional measures are, but this Note points out which success criteria depend on assistive technologies—and therefore would not work by themselves in products with closed functionality.

Because closed functionality, by definition, does not allow a user to attach assistive technology, WCAG success criteria that assume the presence of assistive technology will not facilitate accessibility as WCAG 2.2 intends. Where assistive technologies cannot be used, other output and input solutions are needed to achieve the intent of these success criteria.

Examples of products with closed functionality include:

- an ebook or ebook reader program that allows assistive technologies to access all of the user interface controls of the ebook program (open functionality) but does not allow the assistive technologies to access the actual content of book (closed functionality).
- an operating system that requires the user to provide log in credentials before it allows any assistive technologies to be loaded. The log-in portion would be closed functionality.
- a travel kiosk that provides an audio interface for blind and vision-impaired users as a built-in alternative to the visual interface and tactile keys as an alternative to touch screen operation for both blind users and those who can't operate a touch screen.

See [Appendix A: Success Criteria Problematic for Closed Functionality](#) for a list of success criteria for which this is relevant.

## § 4. Text / Command-line / Terminal Applications and Interfaces

Text applications are a class of software ICT that appeared decades ago, prior to the emergence of the graphical user interface (GUI) and the Web. The interface of a text application is generated using only text characters, and either a hardware terminal or a software terminal application handles the rendering of the text application—similar to how a web user agent handles the rendering of a web application. Text applications only accept text input, though some may also support the use of a mouse or other input devices. More recently, terminal applications which render text applications in the GUI may utilize spoken input through Automated Speech Recognition (ASR). Both GUI and native text environment interfaces also now commonly support word-completion prediction

technologies. Command-line applications are a subset of text applications with further specific properties.

Historically, assistive technologies developed alongside text applications, making it possible for text applications to be accessible. Although there are far fewer new text applications being developed compared to new GUI or web applications, text applications remain in use today. In fact, command-line interfaces have seen a resurgence in recent years, especially in popular programming and revision tracking environments with continued development and greater functionality. In some cases this has precipitated renewed developments in assistive technology support for text applications.

Assistive technology support continues to evolve in today's text applications. Key examples include:

- In command line interfaces (CLI), support often includes context-sensitive help, so that help output following one command argument is different from the help provided following two arguments, and different still after three arguments. This helps users be more efficient and places no new requirements on assistive technologies.
- Output options generally include machine-readable structured text formats (such as JSON), in addition to the still powerful and widely used options of input/output redirection and piping. In these scenarios the assistive technology user can make use of the same range of output options as anyone else who finds the CLI environment compelling.

As noted in [Appendix B. Background on Text / Command-line / Terminal Applications and Interfaces](#), applying WCAG to text / command-line applications involves understanding how text applications are rendered, how text applications have been made accessible via assistive technologies, and how to apply the concepts of “accessibility supported” and “programmatically determined” to text applications.

## § 5. Comments on Conformance

WCAG2ICT is not a standard, so it is not possible to conform to WCAG2ICT. However, some entities may wish to use the information in WCAG2ICT to help establish standards or regulations regarding accessibility in ICT that are based on WCAG 2.2. While such standards or regulations will need to address matters of conformance themselves, the following notes may be of assistance to those wishing to draft their own requirements:

1. The WCAG 2.2 success criteria and the conformance requirements were designed to work together, such that the language of the success criteria is based on the nature of the conformance requirements. The choice of what level to use for a given criteria (A vs. AA vs. AAA) was further influenced by a number of factors specific to the web domain, as set forth in [Understanding Levels of Conformance](#).
2. In the WCAG 2.2 conformance model, a success criteria is satisfied if the item being evaluated does not fail it. If the success criterion is in relation to something that does not exist for the item being evaluated (e.g. a success criterion is about captioning audio and there is no audio) then the success criterion is automatically met. This approach is central to the way the success criteria in WCAG are structured and worded.
3. WCAG 2.2 conformance is applied to the item being evaluated (i.e. web page) as a whole, except when a process includes use of several items, in which case all of the items that are needed in order to complete the process must conform.
4. In WCAG 2.2, when conformance relies on accessibility features of the platform (i.e. browser for web content) or on assistive technologies, WCAG 2.2 requires that there are assistive technologies, etc. that work with the product (web page). That is, conformance with WCAG 2.2 requires that the approaches used are supported by assistive technologies.
5. WCAG 2.2 allows information on part of a page to not conform if the same information is available elsewhere on the page in conforming fashion. However WCAG 2.2 identifies 4 success criteria that must be met on all areas of the page because they can interfere with the user's ability to access and use other parts of the page:
  - [1.4.2 Audio Control](#);
  - [2.1.2 No Keyboard Trap](#);
  - [2.2.2 Pause, Stop, Hide](#).
  - [2.3.1 Three Flashes or Below Threshold](#);

Also, as noted in the Introduction, it wasn't possible to unambiguously carve up software into discrete pieces, and so the unit of evaluation for non-web software is the whole software program. As with any software testing this can be a very large unit of evaluation, and methods similar to standard software testing might be used.

## § 6. Comments by Guideline and Success Criterion

## EDITOR'S NOTE

The WCAG2ICT Task Force has added draft guidance for all of the Level A and AA success criteria that are new in WCAG 2.1. There are placeholders for new WCAG 2.2 level A and AA success criteria, labeled with, "This section is to be developed by the WCAG2ICT Task Force." This guidance will be delivered in a later draft.

The sections that follow are organized according to the principles, guidelines, and success criteria from WCAG 2.2. The text of each item from WCAG 2.2 is copied as quoted text. Following that, the WCAG2ICT guidance is provided. Finally, the "Intent" from Understanding WCAG 2.2 is copied as quoted text; the Task Force makes no substitutions or edits in this text. The WCAG2ICT guidance can be found in the sections where the headings begin with "Guidance When Applying..." to highlight that this is the content specific to this document.

## § 6.1 Perceivable

Information and user interface components must be presentable to users in ways they can perceive.

### § Guidance When Applying Principle 1 to Non-Web Documents and Software

In WCAG 2.2, the Principles are provided for framing and understanding the success criteria under them but are not required for conformance to WCAG. Principle 1 applies directly as written.

#### § 6.1.2 Text Alternatives

Provide text alternatives for any non-text content so that it can be changed into other forms people need, such as large print, braille, speech, symbols or simpler language.

## § *Guidance When Applying Guideline 1.1 to Non-Web Documents and Software*

In WCAG 2.2, the Guidelines are provided for framing and understanding the success criteria under them but are not required for conformance to WCAG. Guideline 1.1 applies directly as written.

### *Intent from Understanding Text Alternatives*

The purpose of this guideline is to ensure that all non-text content is also available in text. "Text" refers to electronic text, not an image of text. Electronic text has the unique advantage that it is presentation neutral. That is, it can be rendered visually, auditorily, tactilely, or by any combination. As a result, information rendered in electronic text can be presented in whatever form best meets the needs of the user. It can also be easily enlarged, spoken aloud so that it is easier for people with reading disabilities to understand, or rendered in whatever tactile form best meets the needs of a user.

*While changing the content into symbols includes changing it into graphic symbols for people with developmental disorders and speech comprehension difficulties, it is not limited to this use of symbols.*

## § 6.1.2.2 *Non-text Content*

All non-text content that is presented to the user has a text alternative that serves the equivalent purpose, except for the situations listed below.

**Controls, Input**

If non-text content is a control or accepts user input, then it has a name that describes its purpose. (Refer to [Success Criterion 4.1.2](#) for additional requirements for controls and content that accepts user input.)

**Time-Based Media**

If non-text content is time-based media, then text alternatives at least provide descriptive identification of the non-text content. (Refer to [Guideline 1.2](#) for additional requirements for media.)

**Test**

If non-text content is a test or exercise that would be invalid if presented in text, then text alternatives at least provide descriptive identification of the non-text content.

**Sensory**

If non-text content is primarily intended to create a specific sensory experience, then text alternatives at least provide descriptive identification of the non-text content.

**CAPTCHA**

If the purpose of non-text content is to confirm that content is being accessed by a person rather than a computer, then text alternatives that identify and describe the purpose of the non-text content are provided, and alternative forms of CAPTCHA using output modes for different types of sensory perception are provided to accommodate different disabilities.

**Decoration, Formatting, Invisible**

If non-text content is pure decoration, is used only for visual formatting, or is not presented to users, then it is implemented in a way that it can be ignored by assistive technology.

## § GUIDANCE WHEN APPLYING SUCCESS CRITERION 1.1.1 TO NON-WEB DOCUMENTS AND SOFTWARE

This applies directly as written, and as described in [Intent from Understanding Success](#)

[Criterion 1.1.1](#) (also provided below).

#### NOTE 1

CAPTCHAs do not currently appear outside of the Web. However, if they do appear, this guidance is accurate.

#### NOTE 2

See also the discussion on [Closed Functionality](#).

*Intent from Understanding Non-text Content*

The intent of this Success Criterion is to make information conveyed by non-text content accessible through the use of a text alternative. Text alternatives are a primary way for making information accessible because they can be rendered through any sensory modality (for example, visual, auditory or tactile) to match the needs of the user. Providing text alternatives allows the information to be rendered in a variety of ways by a variety of user agents. For example, a person who cannot see a picture can have the text alternative read aloud using synthesized speech. A person who cannot hear an audio file can have the text alternative displayed so that he or she can read it. In the future, text alternatives will also allow information to be more easily translated into sign language or into a simpler form of the same language.

## Note on CAPTCHA

CAPTCHAs are a controversial topic in the accessibility community. As is described in the paper [Inaccessibility of CAPTCHA](#), CAPTCHAs intrinsically push the edges of human abilities in an attempt to defeat automated processes. Every type of CAPTCHA will be unsolvable by users with certain disabilities. However, they are widely used, and the Web Content Accessibility Guidelines Working Group believes that if CAPTCHAs were forbidden outright, Web sites would choose not to conform to WCAG rather than abandon CAPTCHA. This would create barriers for a great many more users with disabilities. For this reason the Working Group has chosen to structure the requirement about CAPTCHA in a way that meets the needs of most people with disabilities, yet is also considered adoptable by sites. Requiring two different forms of CAPTCHA on a given site ensures that most people with disabilities will find a form they can use.

Because some users with disabilities will still not be able to access sites that meet the minimum requirements, the Working Group provides recommendations for additional steps. Organizations motivated to conform to WCAG should be aware of the importance of this topic and should go as far beyond the minimum requirements of the guidelines as possible. Additional recommended steps include:

- Providing more than two modalities of CAPTCHAs
- Providing access to a human customer service representative who can bypass CAPTCHA
- Not requiring CAPTCHAs for authorized users

## Additional information

Non-text content can take a number of forms, and this Success Criterion specifies how each is to be handled.

**For non-text content that is not covered by one of the other situations listed below**, such as charts, diagrams, audio recordings, pictures, and animations, text alternatives can make the same information available in a form that can be rendered through any modality (for example, visual, auditory or tactile). Short and long text alternatives can be used as needed to convey the information in the non-text content. Note that **prerecorded audio-only** and **prerecorded video-only** files are covered here. **Live-audio-only** and **Live-video-only** files are covered below (see 3rd paragraph following this one).

**For non-text content that is a control or accepts user input**, such as images used as submit buttons, image maps or complex animations, a name is provided to describe the purpose of the non-text content so that the person at least knows what the non-text content is and why it is there.

**Non-text content that is time-based media** is made accessible through [1.2: Time-Based Media](#). However, it is important that users know what it is when they encounter it on a page so they can decide what action if any they want to take with it. A text alternative that describes the time-based media and/or gives its title is therefore provided.

**For Live Audio-only and live video-only content**, it can be much more difficult to provide text alternatives that provide equivalent information as live audio-only and live video-only content. For these types of non-text content, text alternatives provide a descriptive label.

**Sometimes a test or exercise must be partially or completely presented in non-text format.** Audio or visual information is provided that cannot be changed to text because the test or exercise must be conducted using that sense. For example, a hearing test would be invalid if a text alternative were provided. A visual skill development exercise would similarly make no sense in text form. And a spelling test with text alternatives would not be very effective. For these cases, text alternatives should be provided to describe the purpose of the non-text content; of course, the text alternatives would not provide the same information needed to pass the test.

**Sometimes content is primarily intended to create a specific sensory experience** that words cannot fully capture. Examples include a symphony performance, works of visual art etc. For such content, text alternatives at least identify the non-text content with a descriptive label and where possible, additional descriptive text. If the reason for including the content in the page is known and can be described it is helpful to include

that information.

**Sometimes there are non-text exercises that are used to prove you are human.** To avoid spam robots and other software from gaining access to a site a device called a CAPTCHA is used. These usually involve visual or auditory tasks that are beyond the current capabilities of Web robots. Providing a text alternative to them would however make them operable by Robots, thus defeating their purpose. In this case a text alternative would describe the purpose of the CAPTCHA, and alternate forms using different modalities would be provided to address the needs of people with different disabilities.

**Sometimes there is non-text content that really is not meant to be seen or understood by the user.** Transparent images used to move text over on a page; an invisible image that is used to track usage statistics; and a swirl in the corner that conveys no information but just fills up a blank space to create an aesthetic effect are all examples of this. Putting alternative text on such items just distracts people using screen readers from the content on the page. Not marking the content in any way, though, leaves users guessing what the non-text content is and what information they may have missed (even though they have not missed anything in reality). This type of non-text content, therefore, is marked or implemented in a way that assistive technologies (AT) will ignore it and not present anything to the user.

### § 6.1.3 Time-based Media

Provide alternatives for time-based media.

### § *Guidance When Applying Guideline 1.2 to Non-Web Documents and Software*

In WCAG 2.2, the Guidelines are provided for framing and understanding the success criteria under them but are not required for conformance to WCAG. Guideline 1.2 applies directly as written.

#### *Intent from Understanding Time-based Media*

The purpose of this guideline is to provide access to time-based and synchronized media. This includes media that is:

- audio-only
- video-only
- audio-video
- audio and/or video combined with interaction

To make it easy for authors to quickly determine which success criteria apply to their content, the type of media each success criterion applies to is included in its short name.

For **audio-only** or **video-only** media, you only need to apply the success criteria that say " **audio-only**" or " **video-only**" in their short name. If your media is not **audio-only** or **video-only**, then all the rest of the success criteria apply.

Media can also be **live** or **prerecorded**. Each of the success criterion short names clearly tells you if the success criterion applies to **live** or **prerecorded** media.

Synchronized media is defined in the glossary as:

Note that an audio file accompanied by interaction is covered here, as is a video-only file that involves interaction. These are covered because interaction must take place at a particular time. Having a text transcript that said, "for more information, click now," would not be very helpful since the reader would have no idea when the audio said, "now." As a result, synchronized captions would be needed.

Sometimes, there is so much dialogue that audio description cannot fit into existing pauses in the dialogue. The option at Level A to provide an alternative for time-based media instead of audio description for synchronized media would allow access to all of the information in the synchronized media. This option also allows access to the visual information in non-visual form when audio description is not provided for some other reason.

For synchronized media that includes interaction, interactive elements (for example links) could be embedded in the alternative for time-based media.

This guideline also includes (at Level AAA) sign language interpretation for synchronized media as well as an approach called extended audio description. In extended audio description, the video is frozen periodically to allow more audio

description to take place than is possible in the existing pauses in the dialogue. This is a case where higher-level Success Criteria build upon the requirements of lower-level Success Criterion with the intention of having cumulative, progressively stronger, requirements.

#### § 6.1.3.2 Audio-only and Video-only (Prerecorded)

For prerecorded audio-only and prerecorded video-only media, the following are true, except when the audio or video is a media alternative for text and is clearly labeled as such:

##### **Prerecorded Audio-only**

An alternative for time-based media is provided that presents equivalent information for prerecorded audio-only content.

##### **Prerecorded Video-only**

Either an alternative for time-based media or an audio track is provided that presents equivalent information for prerecorded video-only content.

#### § GUIDANCE WHEN APPLYING SUCCESS CRITERION 1.2.1 TO NON-WEB DOCUMENTS AND SOFTWARE

This applies directly as written, and as described in [Intent from Understanding Success Criterion 1.2.1](#) (also provided below).

##### **NOTE 1**

The alternative can be provided directly in the [non-web document](#) or [software](#) – or provided in an alternate version that meets the success criteria.

##### **NOTE 2**

See also the discussion on [Closed Functionality](#).

### *Intent from Understanding Audio-only and Video-only (Prerecorded)*

The intent of this Success Criterion is to make information conveyed by prerecorded audio-only and prerecorded video-only content available to all users. Alternatives for time-based media that are text based make information accessible because text can be rendered through any sensory modality (for example, visual, auditory or tactile) to match the needs of the user. In the future, text could also be translated into symbols, sign language or simpler forms of the language (future).

An example of pre-recorded video with no audio information or user interaction is a silent movie. The purpose of the transcript is to provide an equivalent to what is presented visually. For prerecorded video content, authors have the option to provide an audio track. The purpose of the audio alternative is to be an equivalent to the video. This makes it possible for users with and without vision impairment to review content simultaneously. The approach can also make it easier for those with cognitive, language and learning disabilities to understand the content because it would provide parallel presentation.

*A text equivalent is not required for audio that is provided as an equivalent for video with no audio information. For example, it is not required to caption video description that is provided as an alternative to a silent movie.*

See also [1.2.4: Audio-only \(Live\)](#)

### § 6.1.3.3 Captions (Prerecorded)

Captions are provided for all prerecorded audio content in synchronized media, except when the media is a media alternative for text and is clearly labeled as such.

### § GUIDANCE WHEN APPLYING SUCCESS CRITERION 1.2.2 TO NON-WEB DOCUMENTS AND SOFTWARE

This applies directly as written, and as described in [Intent from Understanding Success Criterion 1.2.2](#) (also provided below).

## NOTE

The WCAG 2.2 definition of “[captions](#)” notes that “in some countries, captions are called subtitles”. They are also sometimes referred to as “subtitles for the hearing impaired.” Per the definition in WCAG 2.2, to meet this success criterion, whether called captions or subtitles, they would have to provide “synchronized visual and / or text alternative for both speech and non-speech audio information needed to understand the media [content](#)” where non-speech information includes “sound effects, music, laughter, speaker identification and location”.

### *Intent from Understanding Captions (Prerecorded)*

The intent of this Success Criterion is to enable people who are deaf or hard of hearing to watch synchronized media presentations. Captions provide the part of the content available via the audio track. Captions not only include dialogue, but identify who is speaking and include non-speech information conveyed through sound, including meaningful sound effects.

It is acknowledged that at the present time there may be difficulty in creating captions for time-sensitive material and this may result in the author being faced with the choice of delaying the information until captions are available, or publishing time-sensitive content that is inaccessible to the deaf, at least for the interval until captions are available. Over time, the tools for captioning as well as building the captioning into the delivery process can shorten or eliminate such delays.

Captions are not needed when the synchronized media is, itself, an alternate presentation of information that is also presented via text on the Web page. For example, if information on a page is accompanied by a synchronized media presentation that presents no more information than is already presented in text, but is easier for people with cognitive, language, or learning disabilities to understand, then it would not need to be captioned since the information is already presented on the page in text or in text alternatives (e.g., for images).

See also [1.2.4: Captions \(Live\)](#).

### § 6.1.3.4 Audio Description or Media Alternative (Prerecorded)

An alternative for time-based media or audio description of the prerecorded video content is provided for synchronized media, except when the media is a media alternative for text and is clearly labeled as such.

## § GUIDANCE WHEN APPLYING SUCCESS CRITERION 1.2.3 TO NON-WEB DOCUMENTS AND SOFTWARE

This applies directly as written, and as described in [Intent from Understanding Success Criterion 1.2.3](#) (also provided below).

### NOTE 1

The WCAG 2.2 definition of “[audio description](#)” says that “audio description” is “also called ‘video description’ and ‘descriptive narration’”.

### NOTE 2

Secondary or alternate audio tracks are commonly used for this purpose.

### NOTE 3

See also the discussion on [Closed Functionality](#).

*Intent from Understanding Audio Description or Media Alternative (Prerecorded)*

The intent of this Success Criterion is to provide people who are blind or visually impaired access to the visual information in a synchronized media presentation. This Success Criterion describes two approaches, either of which can be used.

One approach is to provide audio description of the video content. The audio description augments the audio portion of the presentation with the information needed when the video portion is not available. During existing pauses in dialogue, audio description provides information about actions, characters, scene changes, and on-screen text that are important and are not described or spoken in the main sound track.

The second approach involves providing all of the information in the synchronized media (both visual and auditory) in text form. An alternative for time-based media provides a running description of all that is going on in the synchronized media content. The alternative for time-based media reads something like a screenplay or book. Unlike audio description, the description of the video portion is not constrained to just the pauses in the existing dialogue. Full descriptions are provided of all visual information, including visual context, actions and expressions of actors, and any other visual material. In addition, non-speech sounds (laughter, off-screen voices, etc.) are described, and transcripts of all dialogue are included. The sequence of description and dialogue transcripts are the same as the sequence in the synchronized media itself. As a result, the alternative for time-based media can provide a much more complete representation of the synchronized media content than audio description alone.

If there is any interaction as part of the synchronized media presentation (e.g., "press now to answer the question") then the alternative for time-based media would provide hyperlinks or whatever is needed to provide the same functionality.

*For 1.2.3, 1.2.5, and 1.2.7, if all of the information in the video track is already provided in the audio track, no audio description is necessary.*

*1.2.3, 1.2.5, and 1.2.8 overlap somewhat with each other. This is to give the author some choice at the minimum conformance level, and to provide additional requirements at higher levels. At Level A in Success Criterion 1.2.3, authors do have the choice of providing either an audio description or a full text alternative. If they wish to conform at Level AA, under Success Criterion 1.2.5 authors must provide an audio description - a requirement already met if they chose that alternative for 1.2.3, otherwise an additional requirement. At Level AAA under Success Criterion 1.2.8 they must provide an extended text description. This is an additional requirement if both 1.2.3 and 1.2.5 were met by providing an audio description only. If 1.2.3 was met, however, by providing a text description, and the 1.2.5 requirement for an audio description was met, then 1.2.8 does not add new requirements.*

See also [1.2.2: Audio Description \(Prerecorded\)](#), [1.2.2: Extended Audio Description \(Prerecorded\)](#) and [1.2.3: Media Alternative \(Prerecorded\)](#).

#### § 6.1.3.5 Captions (Live)

Captions are provided for all live audio content in synchronized media.

#### § GUIDANCE WHEN APPLYING SUCCESS CRITERION 1.2.4 TO NON-WEB DOCUMENTS AND SOFTWARE

This applies directly as written, and as described in [Intent from Understanding Success Criterion 1.2.4](#) (also provided below).

## NOTE

The WCAG 2.2 definition of “[captions](#)” notes that “In some countries, captions are called subtitles”. They are also sometimes referred to as “subtitles for the hearing impaired.” Per the definition in WCAG 2.2, to meet this success criterion, whether called captions or subtitles, they would have to provide “synchronized visual and / or text alternative for both speech and non-speech audio information needed to understand the media [content](#)” where non-speech information includes “sound effects, music, laughter, speaker identification and location”.

### *Intent from Understanding Captions (Live)*

The intent of this Success Criterion is to enable people who are deaf or hard of hearing to watch *real-time* presentations. Captions provide the part of the content available via the audio track. Captions not only include dialogue, but also identify who is speaking and notate sound effects and other significant audio.

This success criterion was intended to apply to broadcast of synchronized media and is not intended to require that two-way multimedia calls between two or more individuals through web apps must be captioned regardless of the needs of users. Responsibility for providing captions would fall to the content providers (the callers) or the “host” caller, and not the application.

### § 6.1.3.6 Audio Description (Prerecorded)

Audio description is provided for all prerecorded video content in synchronized media.

### § GUIDANCE WHEN APPLYING SUCCESS CRITERION 1.2.5 TO NON-WEB DOCUMENTS AND SOFTWARE

This applies directly as written, and as described in [Intent from Understanding Success Criterion 1.2.5](#) (also provided below).

## NOTE 1

The WCAG 2.2 definition of “[audio description](#)” says that audio description is “also called ‘video description’ and ‘descriptive narration’”.

## NOTE 2

Secondary or alternate audio tracks are commonly used for this purpose.

### *Intent from Understanding Audio Description (Prerecorded)*

The intent of this Success Criterion is to provide people who are blind or visually impaired access to the visual information in a synchronized media presentation. The audio description augments the audio portion of the presentation with the information needed when the video portion is not available. During existing pauses in dialogue, audio description provides information about actions, characters, scene changes, and on-screen text that are important and are not described or spoken in the main sound track.

*For 1.2.3, 1.2.5, and 1.2.7, if all of the information in the video track is already provided in the audio track, no audio description is necessary.*

*1.2.3, 1.2.5, and 1.2.8 overlap somewhat with each other. This is to give the author some choice at the minimum conformance level, and to provide additional requirements at higher levels. At Level A in Success Criterion 1.2.3, authors do have the choice of providing either an audio description or a full text alternative. If they wish to conform at Level AA, under Success Criterion 1.2.5 authors must provide an audio description - a requirement already met if they chose that alternative for 1.2.3, otherwise an additional requirement. At Level AAA under Success Criterion 1.2.8 they must provide an extended text description. This is an additional requirement if both 1.2.3 and 1.2.5 were met by providing an audio description only. If 1.2.3 was met, however, by providing a text description, and the 1.2.5 requirement for an audio description was met, then 1.2.8 does not add new requirements.*

## § 6.1.4 Adaptable

Create content that can be presented in different ways (for example simpler layout) without losing information or structure.

### § *Guidance When Applying Guideline 1.3 to Non-Web Documents and Software*

In WCAG 2.2, the Guidelines are provided for framing and understanding the success criteria under them but are not required for conformance to WCAG. Guideline 1.3 applies directly as written.

#### *Intent from Understanding Adaptable*

The purpose of this guideline is to ensure that all information is available in a form that can be perceived by all users, for example, spoken aloud, or presented in a simpler visual layout. If all of the information is available in a form that can be determined by software, then it can be presented to users in different ways (visually, audibly, tactilely etc.). If information is embedded in a particular presentation in such a way that the structure and information cannot be programmatically determined by the assistive technology, then it cannot be rendered in other formats as needed by the user.

The Success Criteria under this guideline all seek to ensure that different types of information that are often encoded in presentation are also available so that they can be presented in other modalities.

- the way the parts of a Web page are organized in relation to each other; and the way a collection of Web pages is organized
- rendering of the content in a form that can be perceived by users

### § 6.1.4.2 Info and Relationships

Information, structure, and relationships conveyed through presentation can be programmatically determined or are available in text.

## § GUIDANCE WHEN APPLYING SUCCESS CRITERION 1.3.1 TO NON-WEB DOCUMENTS AND SOFTWARE

This applies directly as written, and as described in [Intent from Understanding Success Criterion 1.3.1](#) (also provided below).

### NOTE 1

In software, programmatic determinability is best achieved through the use of [accessibility services provided by platform software](#) to enable interoperability between software and assistive technologies and accessibility features of software.

### NOTE 2

See also the discussion on [Closed Functionality](#).

*Intent from Understanding Info and Relationships*

The intent of this Success Criterion is to ensure that information and relationships that are implied by visual or auditory formatting are preserved when the presentation format changes. For example, the presentation format changes when the content is read by a screen reader or when a user style sheet is substituted for the style sheet provided by the author.

Sighted users perceive structure and relationships through various visual cues — headings are often in a larger, bold font separated from paragraphs by blank lines; list items are preceded by a bullet and perhaps indented; paragraphs are separated by a blank line; items that share a common characteristic are organized into tabular rows and columns; form fields may be positioned as groups that share text labels; a different background color may be used to indicate that several items are related to each other; words that have special status are indicated by changing the font family and /or bolding, italicizing, or underlining them; items that share a common characteristic are organized into a table where the relationship of cells sharing the same row or column and the relationship of each cell to its row and/or column header are necessary for understanding; and so on. Having these structures and these relationships programmatically determined or available in text ensures that information important for comprehension will be perceivable to all.

Auditory cues may be used as well. For example, a chime might indicate the beginning of a new section; a change in voice pitch or speech rate may be used to emphasize important information or to indicate quoted text; etc.

When such relationships are perceivable to one set of users, those relationships can be made to be perceivable to all. One method of determining whether or not information has been properly provided to all users is to access the information serially in different modalities.

If links to glossary items are implemented using anchor elements (or the proper link element for the technology in use) and identified using a different font face, a screen reader user will hear that the item is a link when the glossary term is encountered even though they may not receive information about the change in font face. An on-line catalog may indicate prices using a larger font colored red. A screen reader or person who cannot perceive red, still has the information about the price as long as it is preceded by the currency symbol.

Some technologies do not provide a means to programmatically determine some types of information and relationships. In that case then there should be a text description of the information and relationships. For instance, "all required fields are marked with

an asterisk (\*)". The text description should be near the information it is describing (when the page is linearized), such as in the parent element or in the adjacent element.

There may also be cases where it may be a judgment call as to whether the relationships should be programmatically determined or be presented in text. However, when technologies support programmatic relationships, it is strongly encouraged that information and relationships be programmatically determined rather than described in text.

*It is not required that color values be programmatically determined. The information conveyed by color cannot be adequately presented simply by exposing the value. Therefore, [Success Criterion 1.4.1](#) addresses the specific case of color, rather than Success Criterion 1.3.1.*

#### § 6.1.4.3 Meaningful Sequence

When the sequence in which content is presented affects its meaning, a correct reading sequence can be programmatically determined.

#### § GUIDANCE WHEN APPLYING SUCCESS CRITERION 1.3.2 TO NON-WEB DOCUMENTS AND SOFTWARE

This applies directly as written, and as described in [Intent from Understanding Success Criterion 1.3.2](#) (also provided below).

##### NOTE

See also the discussion on [Closed Functionality](#).

*Intent from Understanding Meaningful Sequence*

The intent of this Success Criterion is to enable a user agent to provide an alternative presentation of content while preserving the reading order needed to understand the meaning. It is important that it be possible to programmatically determine at least one sequence of the content that makes sense. Content that does not meet this Success Criterion may confuse or disorient users when assistive technology reads the content in the wrong order, or when alternate style sheets or other formatting changes are applied.

A sequence is *meaningful* if the order of content in the sequence cannot be changed without affecting its meaning. For example, if a page contains two independent articles, the relative order of the articles may not affect their meaning, as long as they are not interleaved. In such a situation, the articles themselves may have meaningful sequence, but the container that contains the articles may not have a meaningful sequence.

The semantics of some elements define whether or not their content is a meaningful sequence. For instance, in HTML, text is always a meaningful sequence. Tables and ordered lists are meaningful sequences, but unordered lists are not.

The order of content in a sequence is not always meaningful. For example, the relative order of the main section of a Web page and a navigation section does not affect their meaning. They could occur in either order in the programmatically determined reading sequence. As another example, a magazine article contains several callout sidebars. The order of the article and the sidebars does not affect their meaning. In these cases there are a number of different reading orders for a Web page that can satisfy the Success Criterion.

For clarity:

1. Providing a particular linear order is only required where it affects meaning.
2. There may be more than one order that is "correct" (according to the WCAG 2.0 definition).
3. Only one correct order needs to be provided.

#### § 6.1.4.4 Sensory Characteristics

Instructions provided for understanding and operating content do not rely solely on sensory characteristics of components such as shape, color, size, visual location, orientation, or sound.

*For requirements related to color, refer to [Guideline 1.4](#).*

## § GUIDANCE WHEN APPLYING SUCCESS CRITERION 1.3.3 TO NON-WEB DOCUMENTS AND SOFTWARE

This applies directly as written, and as described in [Intent from Understanding Success Criterion 1.3.3](#) (also provided below).

*Intent from Understanding Sensory Characteristics*

The intent of this Success Criterion is to ensure that all users can access instructions for using the content, even when they cannot perceive shape or size or use information about spatial location or orientation. Some content relies on knowledge of the shape or position of objects that are not available from the structure of the content (for example, "round button" or "button to the right"). Some users with disabilities are not able to perceive shape or position due to the nature of the assistive technologies they use. This Success Criterion requires that additional information be provided to clarify instructions that are dependent on this kind of information.

Providing information using shape and/or location, however, is an effective method for many users including those with cognitive limitations. This provision should not discourage those types of cues as long as the information is also provided in other ways.

In some languages, it is commonly understood that "above" refers to the content previous to that point in the content and "below" refers to the content after that point. In such languages, if the content being referenced is in the appropriate place in the reading order and the references are unambiguous, statements such as "choose one of the links below" or "all of the above" would conform to this Success Criterion.

WCAG was designed to apply only to controls that were displayed on a web page. The intent was to avoid describing controls solely via references to visual or auditory cues. When applying this to instructions for operating physical hardware controls (e.g. a web kiosk with dedicated content), tactile cues on the hardware might be described (e.g. the arrow shaped key, the round key on the right side). This success criterion is not intended to prevent the use of tactile cues in instructions.

#### § 6.1.4.5 Orientation

Content does not restrict its view and operation to a single display orientation, such as portrait or landscape, unless a specific display orientation is essential.

*Examples where a particular display orientation may be essential are a bank check, a piano application, slides for a projector or television, or virtual reality content where content is not necessarily restricted to landscape or portrait display orientation.*

## § GUIDANCE WHEN APPLYING SUCCESS CRITERION 1.3.4 TO NON-WEB DOCUMENTS AND SOFTWARE

This applies directly as written, and as described in [Intent from Understanding Success Criterion 1.3.4](#) (also provided below).

### NOTE

Content that is only used on hardware with a fixed display orientation OR that has no sensor to detect or change the orientation is covered under the essential exception and not required to provide support for orientation changes.

*Intent from Understanding Orientation*

The intent of this Success Criterion is to ensure that content displays in the orientation (portrait or landscape) preferred by the user. Some websites and applications automatically set and restrict the screen to a particular display orientation and expect that users will respond by rotating their device to match, but this can create problems. Some users have their devices mounted in a fixed orientation (e.g. on the arm of a power wheelchair). Therefore, websites and applications need to support both orientations by not restricting the orientation. Changes in content or functionality due to the size of display are not covered by this criterion which is focused on restrictions of orientation.

Historically, devices tended to have a fixed-orientation display, and all content was created to match that display orientation. Today, most handhelds and many other devices (e.g., monitors) have a hardware-level ability to dynamically adjust default display orientation based on sensor information. The goal of this Success Criterion is that authors should never restrict content's orientation, thus ensuring that it always match the device display orientation.

It is important to distinguish between an author's responsibility not to restrict content to a specific orientation, and device-specific settings governing the locking of display orientation.

Many hand-held devices offer a mechanical switch or a system setting (or both) to allow the user to lock the device display to a specific orientation. Where a user decides to lock their entire device to an orientation, all applications are expected to pick up that setting and to display content accordingly.

This Success Criterion complements device "lock orientation" settings. A web page that does not restrict its display orientation will always support the system-level orientation setting, since the system setting is picked up by the user agent. Alternatively, where a device-level orientation lock is not in place, the user agent should display the page according to the orientation of the device (either its default, or the current orientation determined by any device sensors).

The exception for things considered essential is aimed at situations where the content would only be understood in a particular orientation, or where the technology restricts the possible orientations. If content is aimed at a specific environment which is only available in one orientation (such as a television) then the content can restrict the orientation. Technologies such as virtual reality use screens within goggles that cannot change orientation relative to the user's eyes.

## Benefits

- Users with dexterity impairments, who have a mounted device will be able to use the content in their fixed orientation.
- Users with low-vision will be able to view content in the orientation that works best for them, for example to increase the text size by viewing content in landscape.

### § 6.1.4.6 Identify Input Purpose

The purpose of each input field collecting information about the user can be programmatically determined when:

- The input field serves a purpose identified in the [Input Purposes for user interface components section](#); and
- The content is implemented using technologies with support for identifying the expected meaning for form input data.

### § GUIDANCE WHEN APPLYING SUCCESS CRITERION 1.3.5 TO NON-WEB DOCUMENTS AND SOFTWARE

This applies directly as written, and as described in [Intent from Understanding Success Criterion 1.3.5](#) (also provided below).

#### NOTE 1

[Non-web software](#) and [non-web documents](#) technologies that do not provide attributes that support identifying the expected meaning for the form input data are not in scope for this success criterion.

#### NOTE 2

For non-web software and non-web documents that present input fields, the terms for the input purposes would be the equivalent terms to those listed in the WCAG 2.2 section [Input Purposes for User Interface Components](#) that are supported by the technology used.

#### NOTE 3

See also the discussion on [Closed Functionality](#).

*Intent from Understanding Identify Input Purpose*

The intent of this Success Criterion is to ensure that the purpose of a form input collecting information about the user can be programmatically determined, so that user agents can extract and present this purpose to users using different modalities. The ability to programmatically declare the specific kind of data expected in a particular field makes filling out forms easier, especially for people with cognitive disabilities.

Appropriate visible labels and instruction can help users understand the purpose of form input fields, but users may benefit from having fields that collect specific types of information be rendered in an unambiguous, consistent, and possibly customized way for different modalities - either through defaults in their user agent, or through the aid of assistive technologies.

For some input fields, the type attribute already offers a way to broadly specify the intention of the input field, for example, `input type="tel"`, `input type="email"`, or `input type="password"`. However, these are only very broad categories, describing the type of input, but not necessarily its purpose, especially as it relates to user-specific input fields. As an example, `type="email"` indicates that the field is for an e-mail address but does not clarify if the purpose is for entering the user's e-mail address or some other person's e-mail.

This success criterion defines the types of user interface component input purposes, found in [Section 7 of the WCAG 2.1 Recommendation](#), that must be programmatically identifiable. When these user input purposes are present, and if the technology supports doing so, the field purpose must be programmatically identifiable.

The HTML autocomplete attribute only accepts a certain number of specific well-defined fixed values. This allows a more fine-grained definition or identification of purpose than the type attribute, for example, by allowing the author to specify a specific type of name: Name (`autocomplete="name"`), Given Name (`autocomplete="given-name"`), Family Name (`autocomplete="family-name"`), as well as Username (`autocomplete="username"`), and Nickname (`autocomplete="nickname"`).

By adopting and repurposing this predefined taxonomy of definitions, user agents and assistive technologies can now present the purpose of the inputs to users in different modalities. For example, assistive technologies may display familiar icons next to input fields to help users who have difficulties reading. An icon of a birthday cake may be shown in front of an input field with `autocomplete="bday"`, or the icon of a telephone in front of an input field with `autocomplete="tel"`.

In addition to repurposing this taxonomy, when the autocomplete attribute technique is used to meet this Success Criterion, browsers and other user agents can suggest and 'autofill' the right content by autocompleting these fields based on past user input stored in the browser. By defining more granular definitions of common input purposes, for example "Birthday" (autocomplete="bday"), browsers can store personalized values for each of these fields (the user's birthday date). The user is relieved of having to type the information and can instead confirm or, if needed, change the value of the field, a significant benefit for users with memory issues, dyslexia, and other disabilities. Because the autocomplete values are independent of language, users that may not be familiar with the text used to visually identify user input fields (the label) can still have that purpose consistently identified to them due to the fixed taxonomy of terms.

If an input field accepts two different types of input purpose (as in combined user name/user email fields) and the technology used does not allow multiple purpose values to be defined, it is valid to provide either one or the other value or leave out the designation of input purpose altogether.

When the user agent and assistive technology support for other metadata formats matures, metadata schemes like the [Personalization Semantics Content Module](#) may be used in addition or instead of the HTML autocomplete attribute to identify the purpose of input fields. They can also support automated adaptations that identify and match author-provided input labels to defined vocabularies or symbols that are used instead for labelling inputs.

### § 6.1.5 Distinguishable

Make it easier for users to see and hear content including separating foreground from background.

### § *Guidance When Applying Guideline 1.4 to Non-Web Documents and Software*

In WCAG 2.2, the Guidelines are provided for framing and understanding the success criteria under them but are not required for conformance to WCAG. Guideline 1.4 applies

directly as written.

### *Intent from Understanding Distinguishable*

While some guidelines are focused on making information available in a form that can be presented in alternate formats, this guideline is concerned with making the default presentation as easy to perceive as possible to people with disabilities. The primary focus is on making it easier for users to separate foreground information from the background. For visual presentations this involves making sure that information presented on top of a background contrasts sufficiently with the background. For audio presentations this involves making sure that foreground sounds are sufficiently louder than the background sounds. Individuals with visual and hearing disabilities have much greater difficulty separating foreground and background information.

#### § 6.1.5.2 Use of Color

Color is not used as the only visual means of conveying information, indicating an action, prompting a response, or distinguishing a visual element.

*This success criterion addresses color perception specifically. Other forms of perception are covered in [Guideline 1.3](#) including programmatic access to color and other visual presentation coding.*

#### § GUIDANCE WHEN APPLYING SUCCESS CRITERION 1.4.1 TO NON-WEB DOCUMENTS AND SOFTWARE

This applies directly as written, and as described in [Intent from Understanding Success Criterion 1.4.1](#) (also provided below).

### *Intent from Understanding Use of Color*

The intent of this Success Criterion is to ensure that all sighted users can access information that is conveyed by color differences, that is, by the use of color where each color has a meaning assigned to it. If the information is conveyed through color differences in an image (or other non-text format), the color may not be seen by users with color deficiencies. In this case, providing the information conveyed with color through another visual means ensures users who cannot see color can still perceive the information.

Color is an important asset in design of Web content, enhancing its aesthetic appeal, its usability, and its accessibility. However, some users have difficulty perceiving color. People with partial sight often experience limited color vision, and many older users do not see color well. In addition, people using limited-color or monochrome displays and browsers will be unable to access information that is presented only in color.

Examples of information conveyed by color differences: "required fields are red", "error is shown in red", and "Mary's sales are in red, Tom's are in blue". Examples of indications of an action include: using color to indicate that a link will open in a new window or that a database entry has been updated successfully. An example of prompting a response would be: using highlighting on form fields to indicate that a required field had been left blank.

*This should not in any way discourage the use of color on a page, or even color coding if it is complemented by other visual indication.*

*If content is conveyed through the use of colors that differ not only in their hue, but that also have a significant difference in lightness, then this counts as an additional visual distinction, as long as the difference in relative luminance between the colors leads to a contrast ratio of 3:1 or greater. For example, a light green and a dark red differ **both** by color (hue) **and** by lightness, so they would pass if the contrast ratio is at least 3:1. Similarly, if content is distinguished by inverting an element's foreground and background colors, this would pass (again, assuming that the foreground and background colors have a sufficient contrast).*

*However, if content relies on the user's ability to accurately perceive or differentiate a particular color an additional visual indicator will be required regardless of the contrast ratio between those colors. For example, knowing whether an outline is green for valid or red for invalid.*

*This criterion does not directly address the needs of users with assistive technologies. It aims to ensure that sighted users who cannot distinguish between some colors can still understand content. How information is conveyed to assistive technology users is covered separately in other criteria, including (but not limited to) [1.1.1 Non-text Content](#), [1.3.1 Info and Relationships](#), and [4.1.2 Name, Role, Value](#).*

*Conversely, even if information that is conveyed by color differences is appropriately conveyed to assistive technologies, it does not necessarily pass this criterion, as sighted users who cannot distinguish between certain color may not necessarily be using any assistive technologies. This criterion requires a visible alternative to color.*

Most user agents provide users with a color-only cue that a link has been previously activated by them ("visited"). However, several technical constraints result in authors having very limited control over these color-only indications of visited links. The technical constraints are as follows:

1. User agents constrain the exposure of a link's visited state due to [privacy concerns](#). Author queries to user agents will indicate all links have not been visited.
2. Any available information on the visited state of a link would be inaccurate since it is both user and browser-dependent. Even if an author could accurately get information on previously activated links by a certain user, the author would be constrained to the current browser's preserved history, and would be unable to determine if the user had visited the page using a different browser (or if the history was not preserved, either from cache clearing or use of private sessions).
3. Authors can only use color to modify the `:visited` CSS pseudoclass style. The technology constrains any non-color styling. Due to palette limitations, an author cannot use color alone to achieve 3:1 contrast between link and non-link text as well as between visited and unvisited links while also achieving 4.5:1 contrast for all link and non-link text.
4. Authors also cannot set the visited state of links. The anchor element does not include a "visited" attribute; therefore the author has no ability to alter the state through an attribute setting. As such, authors cannot achieve [1.3.1 Info and Relationships](#) or [4.1.2 Name, Role, Value](#) in regard to visited links.

For these reasons, setting or conveying a link's visited status is not an author responsibility. Where color alone distinguishes between visited and unvisited links, it does not result in a failure of this Success Criterion, even where the contrast between the two link colors is below 3:1. Note that authors must continue to ensure that all text links meet contrast minimums against the page background (SC 1.4.3).

### § 6.1.5.3 Audio Control

If any audio on a Web page plays automatically for more than 3 seconds, either a mechanism is available to pause or stop the audio, or a mechanism is available to control audio volume independently from the overall system volume level.

*Since any content that does not meet this success criterion can interfere with a user's ability to use the whole page, all content on the Web page (whether or not it is used to meet other success criteria) must meet this success criterion. See [Conformance Requirement 5: Non-Interference](#).*

## § GUIDANCE WHEN APPLYING SUCCESS CRITERION 1.4.2 TO NON-WEB DOCUMENTS AND SOFTWARE

This applies directly as written, and as described in [Intent from Understanding Success Criterion 1.4.2](#) (also provided below), replacing “on a Web page” with “in a non-web document or software”, “any content” with “any part of a non-web document or software”, “whole page” with “whole document or software”, “on the Web page” with “in the document or software”, and removing “See Conformance Requirement 5: Non-Interference”.

With these substitutions, it would read:

**1.4.2 Audio Control:** If any audio **[in a non-web document or software]** plays automatically for more than 3 seconds, either a mechanism is available to pause or stop the audio, or a [mechanism](#) is available to control audio volume independently from the overall system volume level. (Level A)

### NOTE

Since any **[part of a non-web document or software]** that does not meet this success criterion can interfere with a user's ability to use the **[whole document or software]**, all [content \[in the document or software\]](#) (whether or not it is used to meet other success criteria) must meet this success criterion.

*Intent from Understanding Audio Control*

Individuals who use screen reading software can find it hard to hear the speech output if there is other audio playing at the same time. This difficulty is exacerbated when the screen reader's speech output is software based (as most are today) and is controlled via the same volume control as the sound. Therefore, it is important that the user be able to turn off the background sound.

Having control of the volume includes being able to reduce its volume to zero. Muting the system volume is not "pausing or stopping" the autoplay audio. Both the "pause or stop" and control of audio volume need to be independent of the overall system volume.

*Playing audio automatically when landing on a page may affect a screen reader user's ability to find the mechanism to stop it because they navigate by listening and automatically started sounds might interfere with that navigation. Therefore, we discourage the practice of automatically starting sounds (especially if they last more than 3 seconds), and encourage that the sound be started by an action initiated by the user after they reach the page, rather than requiring that the sound be stopped by an action of the user after they land on the page.*

See also [1.4.2: Low or No Background Audio](#).

#### § 6.1.5.4 Contrast (Minimum)

The visual presentation of text and images of text has a contrast ratio of at least 4.5:1, except for the following:

##### **Large Text**

Large-scale text and images of large-scale text have a contrast ratio of at least 3:1;

##### **Incidental**

Text or images of text that are part of an inactive user interface component, that are pure decoration, that are not visible to anyone, or that are part of a picture that contains significant other visual content, have no contrast requirement.

##### **Logotypes**

Text that is part of a logo or brand name has no contrast requirement.

## § GUIDANCE WHEN APPLYING SUCCESS CRITERION 1.4.3 TO NON-WEB DOCUMENTS AND SOFTWARE

This applies directly as written, and as described in [Intent from Understanding Success Criterion 1.4.3](#) (also provided below).

*Intent from Understanding Contrast (Minimum)*

The intent of this Success Criterion is to provide enough contrast between text and its background so that it can be read by people with moderately low vision (who do not use contrast-enhancing assistive technology). For people without color deficiencies, hue and saturation have minimal or no effect on legibility as assessed by reading performance (Knoblauch et al., 1991). Color deficiencies can affect luminance contrast somewhat. Therefore, in the recommendation, the contrast is calculated in such a way that color is not a key factor so that people who have a color vision deficit will also have adequate contrast between the text and the background.

Text that is decorative and conveys no information is excluded. For example, if random words are used to create a background and the words could be rearranged or substituted without changing meaning, then it would be decorative and would not need to meet this criterion.

Text that is larger and has wider character strokes is easier to read at lower contrast. The contrast requirement for larger text is therefore lower. This allows authors to use a wider range of color choices for large text, which is helpful for design of pages, particularly titles. 18 point text or 14 point bold text is judged to be large enough to require a lower contrast ratio. (See The American Printing House for the Blind Guidelines for Large Printing and The Library of Congress Guidelines for Large Print under [Resources](#)). "18 point" and "bold" can both have different meanings in different fonts but, except for very thin or unusual fonts, they should be sufficient. Since there are so many different fonts, the general measures are used and a note regarding thin or unusual fonts is included in the definition for large-scale text.

When evaluating this Success Criterion, the font size in points should be obtained from the user agent or calculated on font metrics in the way that user agents do. Point sizes are based on the CSS pt size as defined in [CSS3 Values](#). The ratio between sizes in points and CSS pixels is  $1\text{pt} = 1.333\text{px}$ , therefore 14pt and 18pt are equivalent to approximately 18.5px and 24px.

Because different image editing applications default to different pixel densities (e.g., 72ppi or 96ppi), specifying point sizes for fonts from within an image editing application can be unreliable when it comes to presenting text at a specific size. When creating images of large-scale text, authors should ensure that the text in the resulting image is roughly equivalent to 1.2 and 1.5 em or to 120% or 150% of the default size for body text. For example, for a 72ppi image, an author would need to use approximately 19pt and 24pt font sizes in order to successfully present images of large-scale text to a user.

The 3:1 and 4.5:1 contrast ratios referenced in this Success Criterion are intended to be treated as threshold values. When comparing the computed contrast ratio to the Success Criterion ratio, the computed values should not be rounded (e.g., 4.499:1 would not meet the 4.5:1 threshold).

Because authors do not have control over user settings for font smoothing/anti-aliasing, when evaluating this Success Criterion, refer to the foreground and background colors obtained from the user agent, or the underlying markup and stylesheets, rather than the text as presented on screen.

Due to anti-aliasing, particularly thin or unusual fonts may be rendered by user agents with a much fainter color than the actual text color defined in the underlying CSS. This can lead to situations where text has a contrast ratio that nominally passes the Success Criterion, but has a much lower contrast in practice. In these cases, best practice would be for authors to choose a font with stronger/thicker lines, or to aim for a foreground/background color combination that exceeds the normative requirements of this Success Criterion.

The contrast requirements for text also apply to images of text (text that has been rendered into pixels and then stored in an image format) - see [Success Criterion 1.4.5: Images of Text](#).

This requirement applies to situations in which images of text were intended to be understood as text. Incidental text, such as in photographs that happen to include a

street sign, are not included. Nor is text that for some reason is designed to be invisible to all viewers. Stylized text, such as in corporate logos, should be treated in terms of its function on the page, which may or may not warrant including the content in the text alternative. Corporate visual guidelines beyond logo and logotype are not included in the exception.

In this provision there is an exception that reads "that are part of a picture that contains significant other visual content;". This exception is intended to separate pictures that have text in them from images of text that are done to replace text in order to get a particular look.

*Images of text do not scale as well as text because they tend to pixelate. It is also harder to change foreground and background contrast and color combinations for images of text, which is necessary for some users. Therefore, we suggest using text wherever possible, and when not, consider supplying an image of higher resolution.*

The minimum contrast Success Criterion (1.4.3) applies to text in the page, including placeholder text and text that is shown when a pointer is hovering over an object or when an object has keyboard focus. If any of these are used in a page, the text needs to provide sufficient contrast.

Although this Success Criterion only applies to text, similar issues occur for content presented in charts, graphs, diagrams, and other non-text-based information, which is covered by [Success Criterion 1.4.11 Non-Text Contrast](#).

See also [1.4.6: Contrast \(Enhanced\)](#).

## Rationale for the Ratios Chosen

A contrast ratio of 3:1 is the minimum level recommended by [[ISO-9241-3]] and [[ANSI-HFES-100-1988]] for standard text and vision. The 4.5:1 ratio is used in this provision to account for the loss in contrast that results from moderately low visual acuity, congenital or acquired color deficiencies, or the loss of contrast sensitivity that typically accompanies aging.

The rationale is based on a) adoption of the 3:1 contrast ratio for minimum acceptable contrast for normal observers, in the ANSI standard, and b) the empirical finding that in the population, visual acuity of 20/40 is associated with a contrast sensitivity loss of roughly 1.5 [[ARDITI-FAYE]]. A user with 20/40 would thus require a contrast ratio of 3 \*

1.5 = 4.5 to 1. Following analogous empirical findings and the same logic, the user with 20/80 visual acuity would require contrast of about 7:1.

Hues are perceived differently by users with color vision deficiencies (both congenital and acquired) resulting in different colors and relative luminance contrasts than for normally sighted users. Because of this, effective contrast and readability are different for this population. However, color deficiencies are so diverse that prescribing effective general use color pairs (for contrast) based on quantitative data is not feasible. Requiring good luminance contrast accommodates this by requiring contrast that is independent of color perception. Fortunately, most of the luminance contribution is from the mid and long wave receptors which largely overlap in their spectral responses. The result is that effective luminance contrast can generally be computed without regard to specific color deficiency, except for the use of predominantly long wavelength colors against darker colors (generally appearing black) for those who have protanopia. (We provide an advisory technique on avoiding red on black for that reason). For more information see [[ARDITI-KNOBLAUCH-1994]] [[ARDITI-KNOBLAUCH-1996]] [[ARDITI]].

*Some people with cognitive disabilities require color combinations or hues that have low contrast, and therefore we allow and encourage authors to provide mechanisms to adjust the foreground and background colors of the content. Some of the combinations that could be chosen may have contrast levels that will be lower than those specified here. This is not a violation of this Success Criterion, provided there is a mechanism that will return to the required values set out here.*

The contrast ratio of 4.5:1 was chosen for level AA because it compensated for the loss in contrast sensitivity usually experienced by users with vision loss equivalent to approximately 20/40 vision. (20/40 calculates to approximately 4.5:1.) 20/40 is commonly reported as typical visual acuity of elders at roughly age 80. [[GITTINGS-FOZARD]]

The contrast ratio of 7:1 was chosen for level AAA because it compensated for the loss in contrast sensitivity usually experienced by users with vision loss equivalent to approximately 20/80 vision. People with more than this degree of vision loss usually use assistive technologies to access their content (and the assistive technologies usually have contrast enhancing, as well as magnification capability built into them). The 7:1 level therefore generally provides compensation for the loss in contrast sensitivity experienced by users with low vision who do not use assistive technology and provides contrast enhancement for color deficiency as well.

*Calculations in [\[\[ISO-9241-3\]\]](#) and [\[\[ANSI-HFES-100-1988\]\]](#) are for body text. A relaxed contrast ratio is provided for text that is much larger.*

## Notes on formula

Conversion from nonlinear to linear RGB values is based on IEC/4WD 61966-2-1 [\[\[IEC-4WD\]\]](#).

The formula ( $L1/L2$ ) for contrast is based on [\[\[ISO-9241-3\]\]](#) and [\[\[ANSI-HFES-100-1988\]\]](#) standards.

The ANSI/HFS 100-1988 standard calls for the contribution from ambient light to be included in the calculation of  $L1$  and  $L2$ . The .05 value used is based on Typical Viewing Flare from [\[\[IEC-4WD\]\]](#).

This Success Criterion and its definitions use the terms "contrast ratio" and "relative luminance" rather than "luminance" to reflect the fact that Web content does not emit light itself. The contrast ratio gives a measure of the relative luminance that would result when displayed. (Because it is a ratio, it is dimensionless.)

Refer to [related resources](#) for a list of tools that utilize the contrast ratio to analyze the contrast of Web content.

See also [2.4.7: Focus Visible](#) for techniques for indicating keyboard focus.

## Inactive User Interface Components

User Interface Components that are not available for user interaction (e.g., a disabled control in HTML) are not required to meet contrast requirements. An inactive user interface component is visible but not currently operable. An example would be a submit button at the bottom of a form that is visible but cannot be activated until all the required fields in the form are completed.

Submit Query

*An inactive button using default browser styles*

#### § 6.1.5.5 Resize Text

Except for captions and images of text, text can be resized without assistive technology up to 200 percent without loss of content or functionality.

#### § GUIDANCE WHEN APPLYING SUCCESS CRITERION 1.4.4 TO NON-WEB DOCUMENTS AND SOFTWARE

This applies directly as written, and as described in [Intent from Understanding Success Criterion 1.4.4](#) (also provided below).

##### NOTE 1

[Content](#) for which there are software players, viewers or editors with a 200 percent zoom feature would automatically meet this success criterion when used with such players, unless the content will not work with zoom.

##### NOTE 2

The Intent section refers to the ability to allow users to enlarge the text on screen at least up to 200% without needing to use [assistive technologies](#). This means that the application provides some means for enlarging the text 200% (zoom or otherwise) without loss of [content](#) or functionality or that the application works with the platform features that meet this requirement.

##### NOTE 3

See also the discussion on [Closed Functionality](#).

*Intent from Understanding Resize Text*

The intent of this Success Criterion is to ensure that visually rendered text, including text-based controls (text characters that have been displayed so that they can be seen [vs. text characters that are still in data form such as ASCII]) can be scaled successfully so that it can be read directly by people with mild visual disabilities, without requiring the use of assistive technology such as a screen magnifier. Users may benefit from scaling all content on the Web page, but text is most critical.

The scaling of content is primarily a user agent responsibility. User agents that satisfy [UAAG 1.0 Checkpoint 4.1](#) allow users to configure text scale. The author's responsibility is to create Web content that does not prevent the user agent from scaling the content effectively. Authors may satisfy this Success Criterion by verifying that content does not interfere with user agent support for resizing text, including text-based controls, or by providing direct support for resizing text or changing the layout. An example of direct support might be via server-side script that can be used to assign different style sheets.

The author cannot rely on the user agent to satisfy this Success Criterion for HTML content if users do not have access to a user agent with zoom support. For example, if they work in an environment that requires them to use IE 6.

If the author is using a technology whose user agents do not provide zoom support, the author is responsible to provide this type of functionality directly or to provide content that works with the type of functionality provided by the user agent. If the user agent doesn't provide zoom functionality but does let the user change the text size, the author is responsible for ensuring that the content remains usable when the text is resized.

Some user interface components that function as a label and require activation by the user to access content are not wide enough to accommodate the label's content. For example, in Web mail applications the subject column may not be wide enough to accommodate every possible subject header, but activating the subject header takes the user to the full message with the full subject header. In Web-based spreadsheets, cell content that is too long to be displayed in a column can be truncated, and the full content of the cell is available to the user when the cell receives focus. The content of a user interface component may also become too wide in user interfaces where the user can resize the column width. In this type of user interface component, line wrapping is not required; truncation is acceptable if the component's full content is available on focus or after user activation and an indication that this information can be accessed, is provided to the user in some way besides the fact that it is truncated.

Content satisfies the Success Criterion if it can be scaled up to 200%, that is, up to twice the width and height. Authors may support scaling beyond that limit, however, as scaling becomes more extreme, adaptive layouts may introduce usability problems. For example, words may be too wide to fit into the horizontal space available to them, causing them to be truncated; layout constraints may cause text to overlap with other content when it is scaled larger; or only one word of a sentence may fit on each line, causing the sentence to be displayed as a vertical column of text that is difficult to read.

The working group feels that 200% is a reasonable accommodation that can support a wide range of designs and layouts, and complements older screen magnifiers that provide a minimum magnification of 200%. Above 200%, zoom (which resizes text, images, and layout regions and creates a larger canvas that may require both horizontal and vertical scrolling) may be more effective than text resizing. Assistive technology dedicated to zoom support would usually be used in such a situation and may provide better accessibility than attempts by the author to support the user directly.

*Images of text do not scale as well as text because they tend to pixelate, and therefore we suggest using text wherever possible. It is also harder to change foreground and background contrast and color combinations for images of text, which are necessary for some users.*

See also [1.4.3: Visual Presentation](#).

#### § 6.1.5.6 Images of Text

If the technologies being used can achieve the visual presentation, text is used to convey information rather than images of text except for the following:

##### **Customizable**

The image of text can be visually customized to the user's requirements;

##### **Essential**

A particular presentation of text is essential to the information being conveyed.

*Logotypes (text that is part of a logo or brand name) are considered essential.*

## § GUIDANCE WHEN APPLYING SUCCESS CRITERION 1.4.5 TO NON-WEB DOCUMENTS AND SOFTWARE

This applies directly as written, and as described in [Intent from Understanding Success Criterion 1.4.5](#) (also provided below).

### NOTE

See also the discussion on [Closed Functionality](#).

*Intent from Understanding Images of Text*

The intent of this Success Criterion is to encourage authors, who are using technologies which are capable of achieving their desired default visual presentation, to enable people who require a particular visual presentation of text to be able to adjust the text presentation as needed. This includes people who require the text in a particular font size, foreground and background color, font family, line spacing or alignment.

If an author can use text to achieve the same visual effect, he or she should present the information as text rather than using an image. If for any reason, the author cannot format the text to get the same effect, the effect won't be reliably presented on the commonly available user agents, or using a technology to meet this criterion would interfere with meeting other criteria such as 1.4.4, then an image of text can be used. This includes instances where a particular presentation of text is essential to the information being conveyed, such as type samples, logotypes, branding, etc. Images of text may also be used in order to use a particular font that is either not widely deployed or which the author doesn't have the right to redistribute, or to ensure that the text would be anti-aliased on all user agents.

Images of text can also be used where it is possible for users to customize the image of text to match their requirements.

The definition of image of text contains the note: Note: This does not include text that is part of a picture that contains significant other visual content. Examples of such pictures include graphs, screenshots, and diagrams which visually convey important information through more than just text.

Techniques for satisfying this Success Criterion are the same as those for Success Criterion 1.4.9, except that they only need to apply if the visual presentation can be achieved with the technologies that the author is using. For Success Criterion 1.4.9, the sufficient techniques would be applied only when the user can customize the output.

See also [1.4.9: Images of Text \(No Exception\)](#).

### § 6.1.5.7 Reflow

Content can be presented without loss of information or functionality, and without requiring scrolling in two dimensions for:

- Vertical scrolling content at a width equivalent to 320 CSS pixels;
- Horizontal scrolling content at a height equivalent to 256 CSS pixels.

Except for parts of the content which require two-dimensional layout for usage or meaning.

*320 CSS pixels is equivalent to a starting viewport width of 1280 CSS pixels wide at 400% zoom. For web content which is designed to scroll horizontally (e.g., with vertical text), 256 CSS pixels is equivalent to a starting viewport height of 1024 CSS pixels at 400% zoom.*

*Examples of content which requires two-dimensional layout are images required for understanding (such as maps and diagrams), video, games, presentations, data tables (not individual cells), and interfaces where it is necessary to keep toolbars in view while manipulating content. It is acceptable to provide two-dimensional scrolling for such parts of the content.*

## § GUIDANCE WHEN APPLYING SUCCESS CRITERION 1.4.10 TO NON-WEB DOCUMENTS AND SOFTWARE

## EDITOR'S NOTE

Since there were open issues on WCAG asking about how 1.4.10 Reflow should be applied to non-web software, the WCAG2ICT Task Force would like feedback on whether the guidance for non-web software is sufficient, or if there are other considerations that should be covered in the notes in the WCAG2ICT guidance below. Additionally, the Task Force seeks input and examples for the following:

- Situations where it is unclear whether the exception of a two-dimensional layout might apply.
- Situations where 1.4.10 Reflow cannot be met.
- Are there contexts where text sizing might be an acceptable alternative to 1.4.10 Reflow?
- Are there situations where the content technology and/or platform software do not support reflow? If there are, please explain and indicate whether applications built for this environment should fail this criteria or have an exception so that this requirement doesn't get applied.
- Are there situations where the content technology and/or platform software prevents users from resizing or zooming text, changing the viewport and screen orientation (all cases where reflow of content would typically be needed)? If there are, please explain and indicate whether applications built for this environment should fail this criteria or have an exception so that this requirement is not applied.

This applies directly as written, and as described in [Intent from Understanding Success Criterion 1.4.10](#) (also provided below), replacing “web content” with “content”.

With these substitutions, it would read:

Content can be presented without loss of information or functionality, and without requiring scrolling in two dimensions for:

- Vertical scrolling content at a width equivalent to 320 [CSS pixels](#);
- Horizontal scrolling content at a height equivalent to 256 [CSS pixels](#).

Except for parts of the content which require two-dimensional layout for usage or meaning.

#### NOTE 1

320 CSS pixels is equivalent to a starting viewport width of 1280 CSS pixels wide at 400% zoom. For **[content]** which is designed to scroll horizontally (e.g., with vertical text), 256 CSS pixels is equivalent to a starting viewport height of 1024 CSS pixels at 400% zoom.

#### NOTE 2

Examples of content which requires two-dimensional layout are images required for understanding (such as maps and diagrams), video, games, presentations, data tables (not individual cells), and interfaces where it is necessary to keep toolbars in view while manipulating content. It is acceptable to provide two-dimensional scrolling for such parts of the content.

(non-web documents)

#### NOTE 3

If a [non-web document](#) type and its available [user agents](#) do not support reflow, it may not be possible for a document of that type to meet this success criterion.

(non-web software)

#### NOTE 4

The intent section refers to the ability for content to reflow when user agent zooming is used to scale content or when the [viewport](#) changes in width. For [non-web software](#), this means that when users scale content, adjust the size of a window or dialog, or change the screen resolution, the content will reflow without loss of information or functionality, and without requiring scrolling in two dimensions; or that the application works with platform features to meet this requirement.

#### NOTE 5

Non-web software will have more frequent cases where two-dimensional layout is required for usage or meaning than what occurs on the Web. For example:

- When the software has a complex user interface with toolbars that need to be visible while manipulating content, as explained in the Intent from Understanding 1.4.10 Reflow.

#### NOTE 6

If the content technology and platform software do not support reflow, it may not be possible for non-web software to meet this success criterion, meaning the software application would then fail this success criterion.

#### NOTE 7

Certain platforms do not support adjusting viewports to an equivalent of 320 CSS pixels wide or 256 CSS pixels high. Likewise, some platforms have limitations on zooming as high as 400% for the larger measurements of 1280 CSS pixels wide or 1024 CSS pixels high. In such cases, implement and evaluate at the nearest possible equivalent size to what the Reflow success criterion specifies.

#### NOTE 8

Some software applications provide a mode of operation where reflow is possible, while other modes are unable to reflow. An example is a document authoring tool, which includes both a "print preview mode" (without reflow, for users to view the spatial formatting) and a "drafting view mode" where reflow is supported.

#### NOTE 9

See also the discussion on [Closed Functionality](#).

The intent of this Success Criterion is to support people with low vision who need to enlarge text and read it in a single column. When the browser zoom is used to scale content to 400%, it reflows - i.e., it is presented in one column so that scrolling in more than one direction is not necessary.

For people with low vision, enlarged text with reflow enables reading. It is critical. Enlargement enables perception of characters. Reflow enables tracking. Tracking is following along lines of text, including getting from the end of one line to the beginning of the next line.

Avoiding the need to scroll in the direction of reading in order to reveal lines that are cut off by the viewport is important, because such scrolling significantly increases the effort required to read. It is also important that content is not hidden off-screen. For example, zooming into a vertically scrolling page should not cause content to be hidden to one side.

## How reflow works

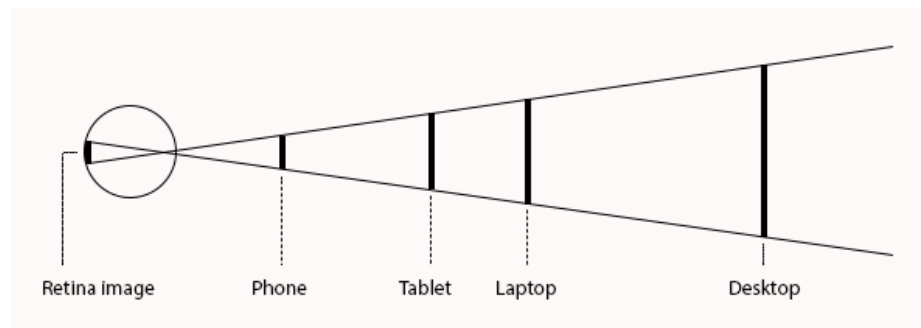
User agents for technologies such as HTML/CSS, PDF, and ePub have methods for reflowing content to fit the width of the window (viewport). When appropriately authored, page content can reflow (wrap) to stay within the window's boundaries (viewport) when users zoom in to enlarge the size of content. Spatial relationships of content may change when users zoom, but all information and functionality should continue to be available.

Supporting the reflow of content is also known as 'Responsive Web Design'. It is enabled by CSS media queries which reformat the web content for different viewport widths (at particular break points) in order to provide optimised layouts for mobile devices such as tablets or smartphones. Importantly, these breakpoints are not only triggered by narrower viewports, but also when users employ the browser zoom function to zoom into the page.

In a desktop browser at 100% (default) scale, typical web pages that support reflow display content in two, three or more columns. Zooming in will at some point trigger a change of layout, so content will now be displayed in fewer columns. At a higher magnification scale of 200% or more, content will usually be rendered in a single column. Parts of content that were in the marginal columns, like a navigation menu or supplementary content, will now typically appear on top of or below the main content.

## Viewing distance and display resolution

The value of 320 CSS pixels was chosen as a reasonable minimum size that authors can achieve. This value lines up with the reported viewport width of small displays of common mobile devices. The width of 320 CSS pixels exactly corresponds to a desktop browser window set to a width of 1280px and zoomed in to 400%. It should be noted that 400% applies to the dimension, not the area. It means four times the default width and four times the default height.



*A letter of the same CSS pixel size on different displays with different resolutions*

When we read, the size of the print is not as important as the image it projects on the retina of our eye. Phones are designed for close viewing while desktops are designed for viewing farther away. As a consequence 16px print on a phone is physically smaller than 16px print on a desktop. This is not a problem because both print sizes cast the same image on our retina if they are viewed at their intended distance.

## Visibility and availability of content

How much of the content is visible may change at different scales. For example, navigation menus that are fully visible in the desktop layout are often collapsed into fewer items, or even into a single menu button (the 'hamburger' icon pattern) so they take up less screen space.

The Success Criterion is met as long as all content and functionality are still fully available - either directly, or revealed via accessible controls, or accessible via direct links.

## Content exceptions for reflow

Content which requires two-dimensional layout for usage or meaning cannot reflow without loss of meaning, and is therefore excepted from the need to be presented without two-dimensional scrolling. For example, graphics and video are by their nature two-dimensional. Cutting up an image and stacking the blocks would render the content unusable. However, it is possible to have these elements stay within the bounds of viewport even as other content zooms to 400% (see advisory techniques).

Data tables have a two-dimensional relationship between the headings and data cells. This relationship is essential to convey the content. This Success Criterion therefore exempts data tables from needing to display without scrolling in the direction of text (e.g., horizontally in a vertically scrolling page). However, cells within data tables are not excepted unless the cell contains types of content that also requires two-dimensional layout for usage or meaning.

Interfaces which provide toolbars to edit content need to show both the content and the toolbar in the viewport. Depending on the number of toolbar buttons, the toolbar may need to scroll in the direction of text.

## Responsive web design and other ways to meet this Success Criterion

Using the responsive web design approach is the most effective method of achieving the goal of allowing people to zoom in to 400%. Each variation (CSS break point) of the page at the same URL should conform (compare [Conformance for WCAG 2.1](#)).

For organisations which are using legacy systems or are not able to update their layout methods for some reason, an alternative conforming version could be a mobile site which has a fixed 320px wide layout. The user should be able to find that version from the default website.

## Avoiding scrolling in horizontally and vertically written languages

The success Criterion applies to both horizontally and vertically written languages. Zooming the page for horizontally written languages where pages scroll vertically by default (e.g. English) should not require horizontal scrolling. Zooming the page for vertically written languages which scroll horizontally by default should not require vertical scrolling.

## The relation of Reflow to the Success Criterion 1.4.4 Resize Text

The focus of the Reflow Success Criterion is to enable users to zoom in without having to scroll in two directions. [Success Criterion 1.4.4 Resize Text](#) also applies, so it should be possible to increase the size of all text to at least 200% while simultaneously meeting the reflow requirement. For most implementations, the text is expected to continue to enlarge as the magnification increases, so that users can magnify text up to (and beyond) 400%. In an implementation where text does not consistently increase its size as people zoom in (such as when it is transformed based on a media query to adapt to small-screen usage), it must still be possible to get to 200% enlargement in order to satisfy the Resize Text criterion.

For example, if at the default browser setting of 100% zoom, text is set at 20px when the window is 1280px wide, the same 20px at 200% zoom would mean a text size of 200%. At 400% zoom, the author may decide to set the text size to 10px: the text would now still be enlarged to 200% compared to the default browser setting of 100% zoom. It is not required to achieve 200% text enlargement at every breakpoint, but it should be possible to get 200% text enlargement in some way.

## Browsers on mobile operating systems

Most browsers on mobile operating systems do not combine reflow and zoom in the same way as on desktop browsers. These mobile browsers normally support reflow when changing the orientation of the device -- content will be adjusted to the new viewport width. However, these mobile browsers can only magnify content to achieve 1.4.4. Resize Text in manners which do not constrain reflow to a single dimension, for example by using a pinch gesture to scale up content or a double tap on a particular column to make it fill the viewport width. This means that zoomed content in most mobile browsers involves two-dimensional scrolling regardless of what an author does.

Mobile user agents *can* offer reflow when users zoom into content, as evidenced by the Dolphin browser for Android. The lack of magnified reflow support in browsers on mobile operating systems can therefore be regarded as a user agent support issue.

The visual presentation of the following have a contrast ratio of at least 3:1 against adjacent color(s):

**User Interface Components**

Visual information required to identify user interface components and states, except for inactive components or where the appearance of the component is determined by the user agent and not modified by the author;

**Graphical Objects**

Parts of graphics required to understand the content, except when a particular presentation of graphics is essential to the information being conveyed.

## § GUIDANCE WHEN APPLYING SUCCESS CRITERION 1.4.11 TO NON-WEB DOCUMENTS AND SOFTWARE

This applies directly as written and as described in [Intent from Understanding Success Criterion 1.4.11](#) (also provided below), replacing "user agent" with "user agent or platform software".

With these substitutions, it would read:

The visual [presentation](#) of the following have a [contrast ratio](#) of at least 3:1 against adjacent color(s):

- **User Interface Components:** Visual information required to identify [user interface components](#) and [states](#), except for inactive components or where the appearance of the component is determined by the **[user agent or platform software]** and not modified by the author;
- **Graphical Objects:** Parts of graphics required to understand the content, except when a particular presentation of graphics is [essential](#) to the information being conveyed.

#### NOTE 1

An example of appearance modification by the author is content that sets the visual style of a control, such as a color or border, to differ from the default style for the user agent or platform.

#### NOTE 2

See also the discussion on [Closed Functionality](#).

*Intent from Understanding Non-text Contrast*

The intent of this Success Criterion is to ensure that active user interface components (i.e., controls) and meaningful graphics are distinguishable by people with moderately low vision. The requirements and rationale are similar to those for large text in [1.4.3 Contrast \(Minimum\)](#).

Low contrast controls are more difficult to perceive, and may be completely missed by people with a visual impairment. Similarly, if a graphic is needed to understand the content or functionality of the webpage then it should be perceivable by people with low vision or other impairments without the need for contrast-enhancing assistive technology.

*The 3:1 contrast ratios referenced in this Success Criterion is intended to be treated as threshold values. When comparing the computed contrast ratio to the Success Criterion ratio, the computed values should not be rounded (e.g. 2.999:1 would not meet the 3:1 threshold).*

## Active User Interface Components

For active controls any visual information provided that is necessary for a user to identify that a control is present and how to operate it must have a minimum 3:1 contrast ratio with the adjacent colors. Also, any visual information necessary to indicate state, such as whether a component is selected or focused must also ensure that the information used to identify the control in that state has a minimum 3:1 contrast ratio.

This Success Criterion does not require that changes in color that differentiate between states of an individual component meet the 3:1 contrast ratio when they do not appear next to each other. For example, there is not a new requirement that visited links contrast with the default color, or that mouse hover indicators contrast with the default state. However, the component must not lose contrast with the adjacent colors, and non-text indicators such as the check in a checkbox, or an arrow graphic indicating a menu is selected or open must have sufficient contrast to the adjacent colors.

### Boundaries

This success criterion does not require that controls have a visual boundary indicating the hit area, but if the visual indicator of the control is the only way to identify the

control, then that indicator must have sufficient contrast. If text (or an icon) within a button or placeholder text inside a text input is visible and there is no visual indication of the hit area then the Success Criterion is passed. If a button with text also has a colored border, since the border does not provide the only indication there is no contrast requirement beyond the text contrast ([1.4.3 Contrast \(Minimum\)](#)). Note that for people with cognitive disabilities it is recommended to delineate the boundary of controls to aid in the recognition of controls and therefore the completion of activities.



*A button (active control) without a visual boundary, and the same button with a focus indicator that is a defined visual boundary of grey (#949494) adjacent to white.*

## Adjacent colors

For user interface components 'adjacent colors' means the colors adjacent to the component. For example, if an input has a white internal background, dark border, and white external background the 'adjacent color' to the component would be the white external background.

Name:

*A standard text input with a grey border (#767676) and white adjacent color outside the component*

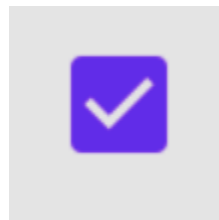
If components use several colors, any color which does not interfere with identifying the component can be ignored for the purpose of measuring contrast ratio. For example, a 3D drop-shadow on an input, or a dark border line between contrasting backgrounds is considered to be subsumed into the color closest in brightness (perceived luminance).

The following example shows an input that has a light background on the inside and a dark background around it. The input also has a dark grey border which is considered to be subsumed into the dark background. The border does not interfere with identifying the component, so the contrast ratio is taken between the white background and dark blue background.

Name:

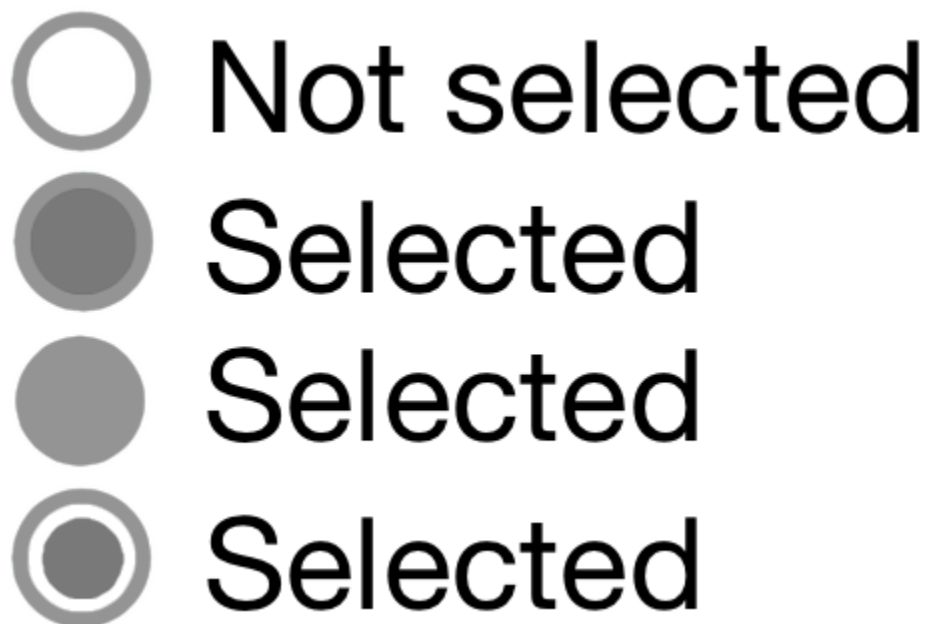
*The contrast of the input background (white) and color adjacent to the control (dark blue #003366) is sufficient. There is also a border (silver) on the component that is not required to contrast with either.*

For visual information required to identify a state, such as the check in a checkbox or the thumb of a slider, that part might be within the component so the adjacent color might be another part of the component.



*A customized checkbox with light grey check (#E5E5E5), which has a contrast ratio of 5.6:1 with the purple box (#6221EA).*

It is possible to use a flat design where the status indicator fills the component and does not contrast with the component, but does contrast with the colors adjacent to the component.



*The first radio button shows the default state with a grey (#949494) circle. The second and third show the radio button selected and filled with a color that contrasts with the color adjacent to the component. The last example shows the state indicator contrasting with the component colors.*

### Relationship with Use of Color

The [Use of Color](#) success criterion addresses changing **only the color** (hue) of an object or text without otherwise altering the object's form. The principle is that contrast ratio (the difference in brightness) can be used to distinguish text or graphics. For example, [G183](#) is a technique to use a contrast ratio of 3:1 with surrounding text to distinguish links and controls. In that case the Working Group regards a link color that meets the 3:1 contrast ratio relative to the non-linked text color as satisfying the Success Criterion [1.4.1 Use of color](#) since it is relying on contrast ratio as well as color (hue) to convey that the text is a link.

Non-text information within controls that uses a change of hue alone to convey the value or state of an input, such as a 1-5 star indicator with a black outline for each star filled with either yellow (full) or white (empty) is likely to fail the Use of color criterion rather than this one.



*Two examples which pass this success criterion, using either a solid fill to indicate a checked-state that has contrast, or a thicker border as well as yellow fill.*



*Two examples which fail a success criterion, the first fails the Use of color criterion due to relying on yellow and white hues. The second example fails the Non-text contrast criterion due to the yellow (#FFF000) to white contrast ratio of 1.2:1.*

Using a change of contrast for focus and other states is a technique to differentiate the states. This is the basis for [G195: Using an author-supplied, highly visible focus indicator](#), and more techniques are being added.

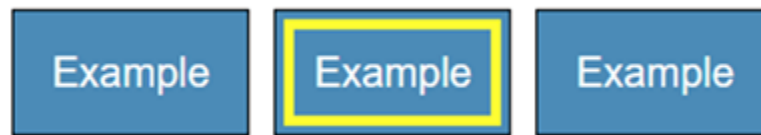
## Relationship with Focus Visible

In combination with [2.4.7 Focus Visible](#), the visual focus indicator for a component *must* have sufficient contrast against the adjacent background when the component is focused, except where the appearance of the component is determined by the user agent and not modified by the author.

Most focus indicators appear outside the component - in that case it needs to contrast with the background that the component is on. Other cases include focus indicators which are:

- only inside the component and need to contrast with the adjacent color(s) within the component.
- the border of the component (inside the component and adjacent to the outside) and need to contrast with both adjacent colours.
- partly inside and partly outside, where either part of the focus indicator can

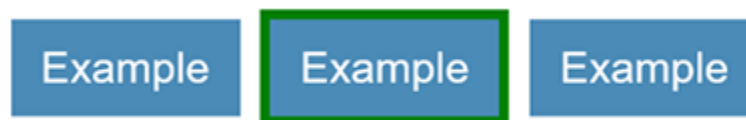
contrast with the adjacent colors.



The internal yellow indicator (#FFFF00) contrasts with the blue button background (#4189B9), **passing** the criterion.



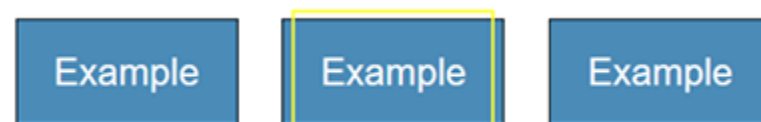
The external yellow indicator (#FFFF00) does not contrast with the white background (#FFF) which the component is on, **failing** the criterion.



The external green indicator (#008000) does contrast with the white background (#FFF) which the component is on, **passing** the criterion. It does not need to contrast with both the component background and the component.

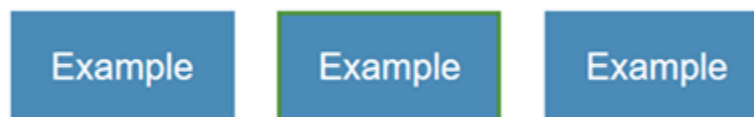
Although the figure above with a dark outline passes non-text contrast, it is not a good indicator unless it is very thick. There is a criterion in WCAG 2.2 that addresses this aspect, [Focus Appearance](#).

If an indicator is partly inside and partly outside the component, either part of the indicator could provide contrast.

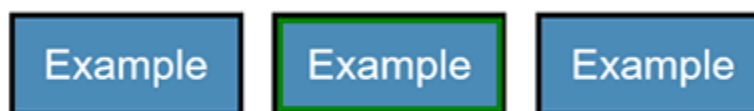


The focus indicator is partially inside, partially outside the button. The internal part of the yellow indicator (#FFFF00) contrasts with the blue button background (#4189B9), **passing** the criterion.

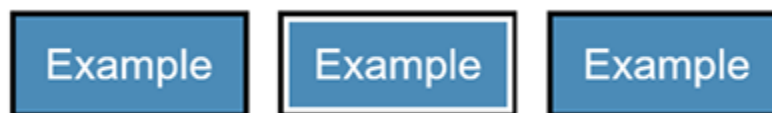
If the focus indicator changes the border of the component within the visible boundary it must contrast with the component. Typically an outline goes around (outside) the visible boundary of the component, in this case changing the border is just inside the visible edge of the component.



*The border of the control changes from blue (#4189B9) to green (#4B933A). This is within the component and does not contrast with the inside background of the component therefore **fails** non-text contrast.*



*An inner border of dark green (#008000) does contrast with the black border, but does not contrast with the blue component background, therefore **fails** non-text contrast.*



*An inner border of white contrasts with the black border and the blue component background, therefore **passes** non-text contrast.*

Note that this Success Criterion does not directly compare the focused and unfocused states of a control - if the focus state relies on a change of color (e.g., changing *only* the background color of a button), this Success Criterion does not define any requirement for the difference in contrast between the two states.



*The change of background within the component is not in scope of non-text contrast. However, this would not pass [Use of color](#).*

## Active User Interface Component Examples

For designing focus indicators, selection indicators and user interface components that need to be perceived clearly, the following are examples that have sufficient contrast.

### Active User Interface Component Examples

Type	Description	Examples
Link Text	Default link text is in the scope of <a href="#">1.4.3 Contrast (Minimum)</a> , and the underline is sufficient to indicate the link.	<a href="#">Link text</a>
Default focus style	Links are required to have a visible focus indicator by <a href="#">2.4.7 Focus Visible</a> . Where the focus style of the user-agent is not adjusted on interactive controls (such as links, form fields or buttons) by the website (author), the default focus style is exempt from contrast requirements (but must still be visible).	<a href="#">Link text</a>
Buttons	A button which has a distinguishing indicator such as position, text style, or context does not need a <i>contrasting</i> visual indicator to show that it is a button, although some users are likely to identify a button with an outline that meets contrast requirements more easily.	Button
Text input (minimal)	Where a text-input has a visual indicator to show it is an input, such as a bottom border (#767676), that indicator must meet 3:1 contrast ratio.	Name: <input type="text"/>
Text input	Where a text-input has an indicator such as a complete border (#767676), that indicator	Name: <input type="text"/>

**Text input  
focus style**

must meet 3:1 contrast ratio.

A focus indicator is required. While in this case the additional gray (#CCC) outline has an insufficient contrast of 1.6:1 against the white (#FFF) background, the cursor/caret which is displayed when the input receives focus *does* provide a sufficiently strong visual indication.

Name:

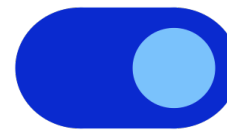
**Text input  
using  
background  
color**

Text inputs that have no border and are differentiated only by a background color must have a 3:1 contrast ratio to the adjacent background (#043464).

Name:

**Toggle  
button**

The toggle button's internal background (#070CD5) has a good contrast with the external white background. Also, the round toggle within (#7AC2FF) contrasts with the internal background.

**Dropdown  
indicator**

The down-arrow is required to understand that there is drop-down functionality, it has a contrast of 4.7:1 for the white icon on dark gray (#6E747B).

Menu ▼

**Dropdown  
indicator**

The down-arrow is required to understand that there is drop-down functionality, it has a contrast of 21:1 for the black icon on white.

Menu ▼

**Checkbox -  
empty**

A black border on a white background indicates the checkbox.

☐ Item

**Checkbox -  
checked**

A black border on a white background indicates the checkbox, the black tick shape

☒ Item

**Checkbox -  
Fail**

indicates the state of checked.

The grey border color of the checkbox (#9D9D9D) has a contrast ratio of 2.7:1 with the white background, which is not sufficient for the visual information required to identify the checkbox.



Item

**Checkbox -  
Subtle hover  
style**

A black border on a white background indicates the checkbox, when the mouse pointer activates the subtle hover state adds a grey background (#DEDEDE). The black border has a 15:1 contrast ratio with the grey background.



Item

**Checkbox -  
Subtle focus  
style - fail**

A focus indicator is required. If the focus indicator is styled by the author, it must meet the 3:1 contrast ratio with adjacent colors. In this case, the gray (#AAA) indicator has an insufficient ratio of 2.3:1 with the white (#FFF) adjacent background.



Item

**Inactive User Interface Components**

User Interface Components that are not available for user interaction (e.g., a disabled control in HTML) are not required to meet contrast requirements. An inactive user interface component is visible but not currently operable. An example would be a submit button at the bottom of a form that is visible but cannot be activated until all the required fields in the form are completed.

Submit Query




*An inactive button using default browser styles*

Inactive components, such as disabled controls in HTML, are not available for user interaction. The decision to exempt inactive controls from the contrast requirements was based on a number of considerations. Although it would be beneficial to some people to discern inactive controls, a one-size-fits-all solution has been very difficult to establish. A method of varying the presentation of disabled controls, such as adding an icon for disabled controls, based on user preferences is anticipated as an advancement in the future.

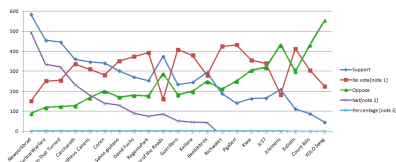
## Graphical Objects

The term "graphical object" applies to stand-alone icons such as a print icon (with no text), and the important parts of a more complex diagram such as each line in a graph. For simple graphics such as single-color icons the entire image is a graphical object. Images made up of multiple lines, colors and shapes will be made of multiple graphical objects, some of which are required for understanding.

Not every graphical object needs to contrast with its surroundings - only those that are required for a user to understand what the graphic is conveying. [Gestalt principles](#) such as the "law of continuity" can be used to ignore minor overlaps with other graphical objects or colors.

Image	Notes
	<p>The phone icon is a simple shape within the orange (#E3660E) circle. The meaning can be understood from that icon alone, the background behind the circle is irrelevant. The orange background and the white icon have a contrast ratio greater than 3:1, which passes.</p> <p>The graphical object is the white phone icon.</p>
	<p>A magnet can be understood by the "U" shape with lighter colored tips. Therefore to understand this graphic you should be able to discern the overall shape (against the background) and the lighter colored tips (against the rest of the U shape and the background).</p> <p>The graphical objects are the "U" shape (by outline or by the solid red color #D0021B), and each tip of the magnet.</p>
	<p>The symbol to show a currency (the £) going down can be understood with recognition of the shape (down arrow) and the currency symbol (pound icon with the shape which is part of the graphic). To understand this graphic you need to discern the arrow shape against the white background, and the pound icon against the yellow background (#F5A623).</p> <p>The graphical objects are the shape and the currency symbol.</p>

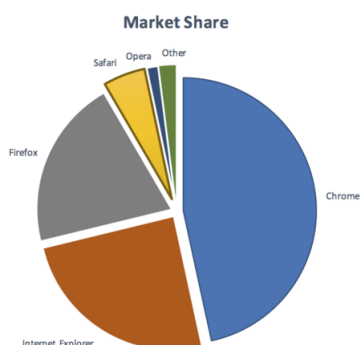
In order to understand the graph you need to discern the lines and shapes for each condition. To



perceive the values of each line along the chart you need to discern the grey lines marking the graduated 100 value increments.

The graphical objects are the lines in the graph, including the background lines for the values, and the colored lines with shapes.

The lines should have 3:1 contrast against their background, but as there is little overlap with other lines they do not need to contrast with each other or the graduated lines. (See the testing principles below.)



To understand the pie chart you have to discern each slice of the pie chart from the others.

The graphical objects are the slices of the pie (chart).

Note: If the values of the pie chart slices were also presented in a conforming manner (see the Pie Charts example for details), the slices would not be required for understanding.

Taking the magnet image above as an example, the process for establishing the graphical object(s) is to:

- Assess what part of each image is needed to understand what it represents. The magnet's "U" shape can be conveyed by the outline or by the red background (either is acceptable). The white tips are also important (otherwise it would be a horseshoe), which needs to contrast with the red background.
- Assume that the user could only see those aspects. Do they contrast with the adjacent colors? The outline of the magnet contrasts with the surrounding text (black/white), and the red and white between the tips also has sufficient contrast.

Due to the strong contrast of the red and white, it would also be possible to only put

the outline around the white tips of the magnet and it would still conform.

## Required for Understanding

The term "required for understanding" is used in the Success Criterion as many graphics do not need to meet the contrast requirements. If a person needs to perceive a graphic, or part of a graphic (a graphical object) in order to understand the content it should have sufficient contrast. However, that is not a requirement when:

- A graphic with text embedded or overlaid conveys the same information, such as labels *and* values on a chart.

*Text within a graphic must meet [1.4.3 Contrast \(Minimum\)](#).*

- The graphic is for aesthetic purposes that does not require the user to see or understand it to understand the content or use the functionality.
- The information is available in another form, such as in a table that follows the graph, which becomes visible when a "Long Description" button is pressed.
- The graphic is part of a logo or brand name (which is considered "essential" to its presentation).

## Gradients

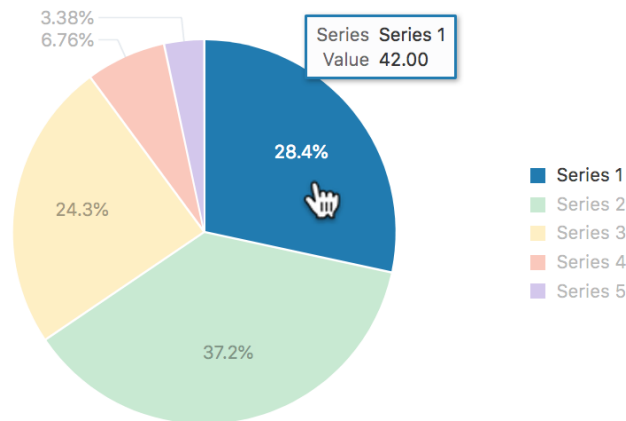
Gradients can reduce the apparent contrast between areas, and make it more difficult to test. The general principle is to identify the graphical object(s) required for understanding, and take the central color of that area. If you remove the adjacent color which does not have sufficient contrast, can you still identify and understand the graphical object?



*Removing the background which does not have sufficient contrast highlights that the graphical object (the "i") is not then understandable.*

## Dynamic Examples

Some graphics may have interactions that either vary the contrast, or display the information as text when you mouseover/tap/focus each graphical object. In order for someone to discern the graphics exist at all, the unfocused default version must already have sufficiently contrasting colors or text. For the area that receives focus, information can then be made available dynamically as pop-up text, or be foregrounded dynamically by increasing the contrast.



*A dynamic chart where the current 'slice' is hovered or focused, which activates the associated text display of the values and highlights the series*

## Infographics

Infographics can mean any graphic conveying data, such as a chart or diagram. On the web it is often used to indicate a large graphic with lots of statements, pictures, charts or other ways of conveying data. In the context of graphics contrast, each item within such an infographic should be treated as a set of graphical objects, regardless of whether it is in one file or separate files.

Infographics often fail to meet several WCAG level AA criteria including:

- [1.1.1 Non-text Content](#)
- [1.4.1 Use of Color](#)
- [1.4.3 \(Text\) Contrast](#)
- [1.4.5 Images of Text](#)

An infographic can use text which meets the other criteria to minimise the number of

graphical objects required for understanding. For example, using text with sufficient contrast to provide the values in a chart. A long description would also be sufficient because then the infograph is not relied upon for understanding.

## Essential Exception

Graphical objects do not have to meet the contrast requirements when "a particular presentation of graphics is essential to the information being conveyed". The Essential exception is intended to apply when there is no way of presenting the graphic with sufficient contrast without undermining the meaning. For example:

- **Logotypes and flags:** The brand logo of an organization or product is the representation of that organization and therefore exempt. Flags may not be identifiable if the colors are changed to have sufficient contrast.
- **Sensory:** There is no requirement to change pictures of real life scenes such as photos of people or scenery.
- **Representing other things:** If you cannot represent the graphic in any other way, it is essential. Examples include:
  - Screenshots to demonstrate how a website appeared.
  - Diagrams of medical information that use the colors found in biology ([example medical schematic from Wikipedia](#)).
  - color gradients that represent a measurement, such as heat maps ([example heatmap from Wikipedia](#)).

## Testing Principles

A summary of the high-level process for finding and assessing non-text graphics on a web page:

- Identify each user-interface component (link, button, form control) on the page and:
  - Identify the visual (non-text) indicators of the component that are required to identify that a control exists, and indicate the current state. In the default (on page load) state, test the contrast ratio against the adjacent colors.
  - Test those contrast indicators in each state.

- Identify each graphic on the page that includes information required for understanding the content (i.e. excluding graphics which have visible text for the same information, or are decorative) and:
  - Check the contrast of the graphical object against its adjacent colors;
  - If there are multiple colors and/or a gradient, choose the least contrasting area to test;
  - If it passes, move to the next graphical object;
  - If the least-contrasting area is less than 3:1, assume that area is invisible, is the graphical object still understandable?
  - If there is enough of the graphical object to understand, it passes, else fail.

The techniques below each have testing criteria, and the related criteria for [Focus visible \(2.4.7\)](#), [Use of color \(1.4.1\)](#), and [Contrast minimum](#) also have techniques.

#### § 6.1.5.9 Text Spacing

In content implemented using markup languages that support the following text style properties, no loss of content or functionality occurs by setting all of the following and by changing no other style property:

- Line height (line spacing) to at least 1.5 times the font size;
- Spacing following paragraphs to at least 2 times the font size;
- Letter spacing (tracking) to at least 0.12 times the font size;
- Word spacing to at least 0.16 times the font size.

Exception: Human languages and scripts that do not make use of one or more of these text style properties in written text can conform using only the properties that exist for that combination of language and script.

#### § GUIDANCE WHEN APPLYING SUCCESS CRITERION 1.4.12 TO NON-WEB DOCUMENTS AND SOFTWARE

This applies directly as written and as described in [Intent from Understanding Success Criterion 1.4.12](#) (also provided below) replacing "In content implemented using markup languages" with "For non-web documents or software content implemented using markup languages" and replacing "that support " with "in a way that supports modification of".

**With these substitutions, it would read:**

**[For non-web documents or software]** content implemented using markup languages **[in a way that supports modification of]** the following [text style properties](#), no loss of content or functionality occurs by setting all of the following and by changing no other style property:

- Line height (line spacing) to at least 1.5 times the font size;
- Spacing following paragraphs to at least 2 times the font size;
- Letter spacing (tracking) to at least 0.12 times the font size;
- Word spacing to at least 0.16 times the font size.

Exception: Human languages and scripts that do not make use of one or more of these text style properties in written text can conform using only the properties that exist for that combination of language and script.

#### NOTE 1

"Content implemented using markup languages" includes parts of software that use markup internally to define a user interface. Examples of markup languages that are used internally to define a software user interface include but are not limited to: HTML (e.g., in [Electron](#) applications or iOS application Web views), XAML, XML (e.g., in Android application layouts), and XUL.

## NOTE 2

There are several mechanisms that allow users to modify text spacing properties of content implemented in markup languages. For example, an eBook technology may have an available user agent that allows users to override document text styles, or a software application may provide a "user style sheet" facility to modify the appearance of the software's own user interface. This Success Criterion does not require that documents and software implement their own mechanisms to allow users to set text spacing; however, when such a mechanism is available, the Success Criterion requires that content respond appropriately to it.

*Intent from Understanding Text Spacing*

The intent of this Success Criterion (SC) is to ensure that when people override author specified text spacing to improve their reading experience, content is still readable and operable. Each of the requirements stipulated in the SC's four bullets helps ensure text styling can be adapted by the user to suit their needs.

The specified metrics set a minimum baseline. The values in between the author's metrics and the metrics specified in this SC should not have loss of content or functionality.

This SC focuses on the adaptability of content to an increase in spacing between lines, words, letters, and paragraphs. Any combination of these may assist a user with effectively reading text. As well, ensuring that content correctly adapts when users override author settings for spacing also significantly increases the likelihood other style preferences can be set by the user. For example, a user may need to change to a wider font family than the author has set in order to effectively read text.

## Author Responsibility

This SC does not dictate that authors must set all their content to the specified metrics. Rather, it specifies that an author's content has the ability to be set to those metrics without loss of content or functionality. The author requirement is both to not interfere with a user's ability to override the author settings, and to ensure that content thus modified does not break content in the manners shown in figures 1 through 3 in Effects of Not Allowing for Spacing Override. The values in the SC are a baseline. Authors are encouraged to surpass the values specified, not see them as a ceiling to build to. If the user chooses to go beyond the metrics specified any resulting loss of content or functionality is the user's responsibility.

Further, this SC is not concerned with *how* users change the line height and spacing metrics. It does not require that content implement its own mechanisms to allow users to do this. It is not a failure of the content if a user agent or platform does not provide a way for users to do this. Content does not fail this SC if the method chosen by the user - for instance, the use of an extension or bookmarklet - fails to correctly set the line height and spacing text properties on the content (provided that the content is not actively and purposely preventing the properties from being added).

## Applicability

If the markup-based technologies being used are capable of overriding text to the Success Criterion's metrics, then this SC is applicable. For instance Cascading Style Sheet/HTML technologies are quite able to allow for the specified spacing metrics. Plugin technologies would need to have a built-in ability to modify styles to the specified metrics. Currently, this SC does not apply to PDF as it is not implemented using markup.

Examples of text that are typically not affected by style properties and not expected to adapt are:

- Video captions embedded directly into the video frames and not provided as an associated caption file
- Images of text

For this SC, [canvas](#) implementations of text are considered to be images of text.

## Use of ellipses

There may be regions of a page where text containers cannot expand due to design constraints (such as a maximum width for the left navigation or table column headers). A common convention if text exceeds its space is to replace truncated text with an ellipsis. Where ellipses appear as a result of modifying text style properties, the page can still meet the Text Spacing requirements, so long as the content is still available. For example:

- a mechanism is provided to reveal the truncated text on the page (for instance, the text appears on focus or on activation)
- where the ellipsis is part of a section of content which includes a link, the truncated text is revealed on the linked page

Where text is not truncated but it is when text is spaced, if there is no mechanism to show the truncated text, it fails this Success Criterion.

## User Responsibility

The ability to read and derive meaning from the overridden spacing rests with the user. The user may choose to exceed the spacing adjustments in the SC. If the increased spacing causes loss of content or functionality, the user will adjust or return to the

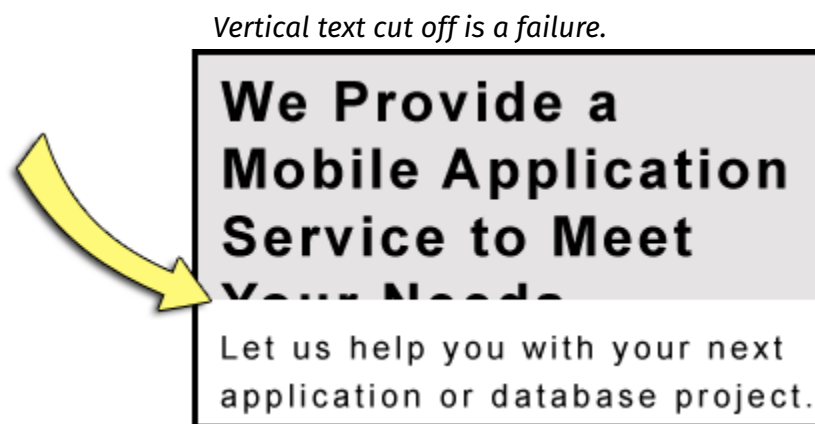
author's original spacing or spacing within the bounds of the SC. Regardless, the user needs the flexibility to adjust spacing within the bounds set in the SC without loss of content or functionality. Such changes may be achieved via user stylesheet, bookmarklet, extension, or application.

## Effects of Not Allowing for Spacing Override

The following images show some types of failures when authors do not take into consideration that users may override spacing to the metrics specified in this Success Criterion.

### Text Cut Off

The bottom portion of the words "Your Needs" is cut off in a heading making that text unreadable in Figure 1. It should read "We Provide a Mobile Application Service to Meet Your Needs."



In Figure 2 the last portion of text is cut off in 3 side-by-side headings. The 1st heading should read "A cog in the wheel." But it reads "A cog in the whe". Only half of the second "e" is visible and the letter "l" is completely missing. The 2nd heading should read "A penny for your thoughts". But it reads "A penny for your". The 3rd should read "Back to the drawing board." But it reads "Back to the drawi".

*Horizontal text cut off is a failure.*



## Text Overlap

In Figure 3 the last 3 words "Groups and Programs" of the heading "Technologists Seeking Input from Groups and Programs" overlap the following sentence. That sentence should read, "You are invited to share ideas and areas of interest related to the integration of technology from a group or program perspective." But the words "You are invited to share ideas" are obscured and unreadable.

*Overlapping text is a failure.*



### § 6.1.5.10 Content on Hover or Focus

Where receiving and then removing pointer hover or keyboard focus triggers additional content to become visible and then hidden, the following are true:

**Dismissible**

A mechanism is available to dismiss the additional content without moving pointer hover or keyboard focus, unless the additional content communicates an input error or does not obscure or replace other content;

**Hoverable**

If pointer hover can trigger the additional content, then the pointer can be moved over the additional content without the additional content disappearing;

**Persistent**

The additional content remains visible until the hover or focus trigger is removed, the user dismisses it, or its information is no longer valid.

Exception: The visual presentation of the additional content is controlled by the user agent and is not modified by the author.

*Examples of additional content controlled by the user agent include browser tooltips created through use of the HTML [title attribute](#).*

*Custom tooltips, sub-menus, and other nonmodal popups that display on hover and focus are examples of additional content covered by this criterion.*

*This criterion applies to content that appears in addition to the triggering component itself. Since hidden components that are made visible on keyboard focus (such as links used to skip to another part of a page) do not present additional content they are not covered by this criterion.*

## § GUIDANCE WHEN APPLYING SUCCESS CRITERION 1.4.13 TO NON-WEB DOCUMENTS AND SOFTWARE

This applies directly as written, and as described in [Intent from Understanding Success Criterion 1.4.13](#) (also provided below), replacing "user agent" with "user agent or platform software", "browser tooltips" with "tooltips", and "the HTML title attribute" with "user interface object attributes".

With these substitutions, it would read:

Where receiving and then removing pointer hover or keyboard focus triggers additional content to become visible and then hidden, the following are true:

**Dismissible** A [mechanism](#) is available to dismiss the additional content without moving pointer hover or keyboard focus, unless the additional content communicates an [input error](#) or does not obscure or replace other content;

**Hoverable** If pointer hover can trigger the additional content, then the pointer can be moved over the additional content without the additional content disappearing;

**Persistent** The additional content remains visible until the hover or focus trigger is removed, the user dismisses it, or its information is no longer valid.

Exception: The visual presentation of the additional content is controlled by the [\[user agent or platform software\]](#) and is not modified by the author.

#### NOTE 1

Examples of additional content controlled by the [\[user agent or platform software\]](#) include [\[tooltips\]](#) created through use of [\[user interface object attributes\]](#).

#### NOTE 2

Custom tooltips, sub-menus, and other nonmodal popups that display on hover and focus are examples of additional content covered by this criterion.

#### NOTE 3

##### NOTE

This criterion applies to content that appears in addition to the triggering component itself. Since hidden components that are made visible on keyboard focus (such as links used to skip to another part of a page) do not present additional content they are not covered by this criterion.

*Intent from Understanding Content on Hover or Focus*

Additional content that appears and disappears in coordination with keyboard focus or pointer hover often leads to accessibility issues. Reasons for such issues include:

1. the user may not have intended to trigger the interaction
2. the user may not know new content has appeared
3. the new content may interfere with a user's ability to do a task

Examples of such interactions can include custom tooltips, sub-menus and other nonmodal popups which display on hover and focus. The intent of this success criterion is to ensure that authors who cause additional content to appear and disappear in this manner must design the interaction in such a way that users can:

- perceive the additional content AND
- dismiss it without disrupting their page experience.

There are usually more predictable and accessible means of adding content to the page, which authors are recommended to employ. If an author *does* choose to make additional content appear and disappear in coordination with hover and keyboard focus, this success criterion specifies three conditions that must be met:

- dismissable
- hoverable
- persistent

Each of these is discussed in a separate section.

## Dismissable

The intent of this condition is to ensure that the additional content does not interfere with viewing or operating the page's original content. When magnified, the portion of the page visible in the viewport can be significantly reduced. Mouse users frequently move the pointer to pan the magnified viewport and display another portion of the screen. However, almost the entire portion of the page visible in this restricted viewport may trigger the additional content, making it difficult for a user to pan without re-triggering the content. A keyboard means of dismissing the additional content provides a workaround.

Alternatively, low vision users who can only navigate via the keyboard do not want the

small area of their magnified viewport cluttered with hover text. They need a keyboard method of dismissing something that is obscuring the current focal area.

Two methods may be used to satisfy this condition and prevent such interference:

1. Position the additional content so that it does not obscure any other content including the trigger, with the exception of white space and purely decorative content, such as a background graphic which provides no information.
2. Provide a mechanism to easily dismiss the additional content, such as by pressing Escape.

For most triggers of relatively small size, it is desirable for both methods to be implemented. If the trigger is large, noticing the additional content may be of concern if it appears away from the trigger. In those cases, only the second method may be appropriate.

The success criterion allows for input error messages to persist as there are cases that require attention, explicit confirmation or remedial action.

## Hoverable

The intent of this condition is to ensure that additional content which may appear on hover of a target may also be hovered itself. Content which appears on hover can be difficult or impossible to perceive if a user is required to keep their mouse pointer over the trigger. When the added content is large, magnified views may mean that the user needs to scroll or pan to completely view it, which is impossible unless the user is able to move their pointer off the trigger without the additional content disappearing.

Another common situation is when large pointers have been selected via platform settings or assistive technology. Here, the pointer can obscure a significant area of the additional content. A technique to view the content fully in both situations is to move the mouse pointer directly from the trigger onto the new content. This capability also offers significant advantages for users who utilize screen reader feedback on mouse interactions. This condition generally implies that the additional content overlaps or is positioned adjacent to the target.

## Persistent

The intent of this condition is to ensure users have adequate time to perceive the

additional content after it becomes visible. Users with disabilities may require more time for many reasons, such as to change magnification, move the pointer, or simply to bring the new content into their visual field. Once it appears, the content should remain visible until:

- The user removes hover or focus from the trigger and the additional content, consistent with the typical user experience;
- The user dismisses the additional content via the mechanism provided to satisfy the Dismissible condition; or
- The information conveyed by the additional content becomes invalid, such as a 'busy' message that is no longer valid.

## Additional Notes

- This criterion does not attempt to solve such issues when the appearance of the additional content is completely controlled by the user agent. A prominent example is the common behavior of browsers to display the `title` attribute in HTML as a small tooltip.
- Modal dialogs are out of scope for this criterion because they must take keyboard focus and thus should not appear on hover or focus. Refer to [Success Criterion 3.2.1, On Focus](#).
- Content which can be triggered via pointer hover should also be able to be triggered by keyboard focus. Refer to [Success Criterion 2.1.1, Keyboard](#).

## § 6.2 Operable

User interface components and navigation must be operable.

## § Guidance When Applying Principle 2 to Non-Web Documents and Software

In WCAG 2.2, the Principles are provided for framing and understanding the success criteria

under them but are not required for conformance to WCAG. Principle 2 applies directly as written.

## § 6.2.2 Keyboard Accessible

Make all functionality available from a keyboard.

### § *Guidance When Applying Guideline 2.1 to Non-Web Documents and Software*

In WCAG 2.2, the Guidelines are provided for framing and understanding the success criteria under them but are not required for conformance to WCAG. Guideline 2.1 applies directly as written.

#### *Intent from Understanding Keyboard Accessible*

If all functionality can be achieved using the keyboard, it can be accomplished by keyboard users, by speech input (which creates keyboard input), by mouse (using on-screen keyboards), and by a wide variety of assistive technologies that create simulated keystrokes as their output. No other input form has this flexibility or is universally supported and operable by people with different disabilities, as long as the keyboard input is not time-dependent.

Note that providing universal keyboard input does not mean that other types of input should not be supported. Optimized speech input, optimized mouse/pointer input, etc., are also good. The key is to provide keyboard input and control as well.

Some devices do not have native keyboards—for example, a PDA or cell phone. If these devices have a Web browsing capability, however, they will have some means of generating text or "keystrokes". This guideline uses the term "keyboard interface" to acknowledge that Web content should be controlled from keystrokes that may come from a keyboard, keyboard emulator, or other hardware or software that generates keyboard or text input.

### § 6.2.2.2 Keyboard

All functionality of the content is operable through a keyboard interface without requiring specific timings for individual keystrokes, except where the underlying function requires input that depends on the path of the user's movement and not just the endpoints.

*This exception relates to the underlying function, not the input technique. For example, if using handwriting to enter text, the input technique (handwriting) requires path-dependent input but the underlying function (text input) does not.*

*This does not forbid and should not discourage providing mouse input or other input methods in addition to keyboard operation.*

### § GUIDANCE WHEN APPLYING SUCCESS CRITERION 2.1.1 TO NON-WEB DOCUMENTS AND SOFTWARE

This applies directly as written, and as described in [Intent from Understanding Success Criterion 2.1.1](#) (also provided below).

#### NOTE 1

This does not imply that software always needs to directly support a keyboard or “keyboard interface”. Nor does it imply that software always needs to provide a soft keyboard. Underlying platform software may provide device independent input services to applications that enable operation via a keyboard. Software that supports operation via such platform device independent services would be operable by a keyboard and would comply.

#### NOTE 2

See also the discussion on [Closed Functionality](#).

## *Intent from Understanding Keyboard*

The intent of this Success Criterion is to ensure that, wherever possible, content can be operated through a keyboard or keyboard interface (so an alternate keyboard can be used). When content can be operated through a keyboard or alternate keyboard, it is operable by people with no vision (who cannot use devices such as mice that require eye-hand coordination) as well as by people who must use alternate keyboards or input devices that act as keyboard emulators. Keyboard emulators include speech input software, sip-and-puff software, on-screen keyboards, scanning software and a variety of assistive technologies and alternate keyboards. Individuals with low vision also may have trouble tracking a pointer and find the use of software much easier (or only possible) if they can control it from the keyboard.

Examples of "specific timings for individual keystrokes" include situations where a user would be required to repeat or execute multiple keystrokes within a short period of time or where a key must be held down for an extended period before the keystroke is registered.

The phrase "except where the underlying function requires input that depends on the path of the user's movement and not just the endpoints" is included to separate those things that cannot reasonably be controlled from a keyboard.

Most actions carried out by a pointing device can also be done from the keyboard (for example, clicking, selecting, moving, sizing). However, there is a small class of input that is done with a pointing device that cannot be done from the keyboard in any known fashion without requiring an inordinate number of keystrokes. Free hand drawing, watercolor painting, and flying a helicopter through an obstacle course are all examples of functions that require path dependent input. Drawing straight lines, regular geometric shapes, re-sizing windows and dragging objects to a location (when the path to that location is not relevant) do not require path dependent input.

The use of MouseKeys would not satisfy this Success Criterion because it is not a keyboard equivalent to the application; it is a mouse equivalent (i.e., it looks like a mouse to the application).

It is assumed that the design of user input features takes into account that operating system keyboard accessibility features may be in use. For example, modifier key locking may be turned on. Content continues to function in such an environment, not sending events that would collide with the modifier key lock to produce unexpected results.

### § 6.2.2.3 No Keyboard Trap

If keyboard focus can be moved to a component of the page using a keyboard interface, then focus can be moved away from that component using only a keyboard interface, and, if it requires more than unmodified arrow or tab keys or other standard exit methods, the user is advised of the method for moving focus away.

*Since any content that does not meet this success criterion can interfere with a user's ability to use the whole page, all content on the Web page (whether it is used to meet other success criteria or not) must meet this success criterion. See [Conformance Requirement 5: Non-Interference](#).*

## § GUIDANCE WHEN APPLYING SUCCESS CRITERION 2.1.2 TO NON-WEB DOCUMENTS AND SOFTWARE

This applies directly as written, and as described in [Intent from Understanding Success Criterion 2.1.2](#) (also provided below), replacing “page” and “Web page” with “non-web document or software” and removing “See Conformance Requirement 5: Non-Interference”.

With these substitutions, it would read:

**2.1.2 No Keyboard Trap:** If keyboard focus can be moved to a component of the **[non-web document or software]** using a [keyboard interface](#), then focus can be moved away from that component using only a keyboard interface, and, if it requires more than unmodified arrow or tab keys or other standard exit methods, the user is advised of the method for moving focus away. (Level A)

### NOTE 1

Since any content that does not meet this success criterion can interfere with a user's ability to use the whole **[non-web document or software]**, all content on the **[non-web document or software]** (whether it is used to meet other success criteria or not) must meet this success criterion.

## NOTE 2

Standard exit methods may vary by platform. For example, on many desktop platforms, the Escape key is a standard method for exiting.

### *Intent from Understanding No Keyboard Trap*

The intent of this Success Criterion is to ensure that that content does not "trap" keyboard focus within subsections of content on a Web page. This is a common problem when multiple formats are combined within a page and rendered using plug-ins or embedded applications.

There may be times when the functionality of the Web page restricts the focus to a subsection of the content, as long as the user knows how to leave that state and "untrap" the focus.

### § 6.2.2.4 Character Key Shortcuts

If a keyboard shortcut is implemented in content using only letter (including upper- and lower-case letters), punctuation, number, or symbol characters, then at least one of the following is true:

#### **Turn off**

A mechanism is available to turn the shortcut off;

#### **Remap**

A mechanism is available to remap the shortcut to include one or more non-printable keyboard keys (e.g., Ctrl, Alt);

#### **Active only on focus**

The keyboard shortcut for a user interface component is only active when that component has focus.

### § GUIDANCE WHEN APPLYING SUCCESS CRITERION 2.1.4 TO NON-WEB DOCUMENTS AND SOFTWARE

This applies directly as written, and as described in [Intent from Understanding Success Criterion 2.1.4](#) (also provided below).

#### NOTE

The WCAG2ICT interpretation is that a long press of a key (2 seconds or more) and other accessibility features provided by the platform do not meet the WCAG definition of a keyboard shortcut. See the [keyboard shortcut](#) definition for more details.

*Intent from Understanding Character Key Shortcuts*

The intent of this Success Criterion is to reduce accidental activation of keyboard shortcuts. Character key shortcuts work well for many keyboard users, but are inappropriate and frustrating for speech input users — whose means of input is strings of letters — and for keyboard users who are prone to accidentally hit keys. To rectify this issue, authors need to allow users to turn off or reconfigure shortcuts that are made up of only character keys.

Note that this success criterion doesn't affect components such as listboxes and drop-down menus. Although these components contain values (words) that may be selected by one or more character keys, the shortcuts are only active when the components have focus. Other components such as menus may be accessed or opened with a single non-character shortcut (e.g., Alt or Alt+F) before pressing a single character key to select an item. This makes the full path to invoking a menu a two-step shortcut that includes a non-printable key. [Accesskeys](#) are also not affected because they include modifier keys.

Speech Input users generally work in a single mode where they can use a mix of dictation and speech commands. This works well because the user knows to pause before and after commands, and commands are usually at least two words long. So, for instance, a user might say a bit of dictation, such as "the small boat", then pause, and say a command to delete that dictation, such as "Delete Line". In contrast, if the user were to say the two phrases together without a pause, the whole phrase would come out as dictation (i.e., "the small boat delete line"). Although speech input programs often include modes that listen only for dictation or only for commands, most speech users use the all-encompassing mode all the time because it is a much more efficient workflow. It could decrease command efficiency significantly if users were to change to command mode and back before and after issuing each command.

Speech users can also speak most keyboard commands (e.g., "press Control Foxtrot") without any problems. If the website or app is keyboard enabled, the speech user can also write a native speech macro that calls the keyboard command, such as "This Print" to carry out Ctrl+P.

Single-key shortcuts are the exception. While using single letter keys as controls might be appropriate and efficient for many keyboard users, single-key shortcuts are disastrous for speech users. The reason for this is that when only a single key is used to trip a command, a spoken word can become a barrage of single-key commands if the cursor focus happens to be in the wrong place.

For example, a speech-input user named Kim has her cursor focus in the main window

of a web mail application that uses common keyboard shortcuts to navigate ("k"), archive ("y") and mute messages ("m"). A coworker named Mike enters her office and says "Hey Kim" and her microphone picks that up. The Y of "hey" archives the current message. K in "Kim" moves down one conversation and M mutes a message or thread. And, if Kim looks up and says "Hey Mike" without remembering to turn off the microphone, the same three things happen in a different sequence.

A user interacting with a webpage or web app that doesn't use single-character shortcuts doesn't have this problem. Inadvertent strings of characters from the speech application are not interpreted as shortcuts if a modifier key is required. A speech user filling in a text input form may find that a phrase that is accidentally picked up by the speech microphone results in stray text being entered into the field, but that is easily seen and undone. The Resources section of this page contains links to videos demonstrating these types of issues.

## Benefits

- Speech users will be able to turn off single-key shortcuts so they can avoid accidentally firing batches of them at once. This will allow speech users to make full use of programs that offer single-key shortcuts to keyboard users.
- Keyboard-only users who have dexterity challenges can also be prone to accidentally hitting keys. Those users would be able to avoid problematic single character shortcuts by turning them off or modifying them to include at least one non-character key.
- Allowing *all* shortcut keys to be remapped can help users with some cognitive disabilities, since the same shortcuts can be assigned to perform the same actions across different applications.

### § 6.2.3 Enough Time

Provide users enough time to read and use content.

## § *Guidance When Applying Guideline 2.2 to Non-Web Documents and Software*

In WCAG 2.2, the Guidelines are provided for framing and understanding the success criteria under them but are not required for conformance to WCAG. Guideline 2.2 applies directly as written.

### *Intent from Understanding Enough Time*

Many users who have disabilities need more time to complete tasks than the majority of users: they may take longer to physically respond, they may take longer to read things, they may have low vision and take longer to find things or to read them, or they may be accessing content through an assistive technology that requires more time. This guideline focuses on ensuring that users are able to complete the tasks required by the content with their own individual response times. The primary approaches deal with eliminating time constraints or providing users enough additional time to allow them to complete their tasks. Exceptions are provided for those cases where this is not possible.

### § 6.2.3.2 *Timing Adjustable*

For each time limit that is set by the content, at least one of the following is true:

**Turn off**

The user is allowed to turn off the time limit before encountering it; or

**Adjust**

The user is allowed to adjust the time limit before encountering it over a wide range that is at least ten times the length of the default setting; or

**Extend**

The user is warned before time expires and given at least 20 seconds to extend the time limit with a simple action (for example, "press the space bar"), and the user is allowed to extend the time limit at least ten times; or

**Real-time Exception**

The time limit is a required part of a real-time event (for example, an auction), and no alternative to the time limit is possible; or

**Essential Exception**

The time limit is essential and extending it would invalidate the activity; or

**20 Hour Exception**

The time limit is longer than 20 hours.

*This success criterion helps ensure that users can complete tasks without unexpected changes in content or context that are a result of a time limit. This success criterion should be considered in conjunction with [Success Criterion 3.2.1](#), which puts limits on changes of content or context as a result of user action.*

## § GUIDANCE WHEN APPLYING SUCCESS CRITERION 2.2.1 TO NON-WEB DOCUMENTS AND SOFTWARE

This applies directly as written, and as described in [Intent from Understanding Success Criterion 2.2.1](#) (also provided below), replacing “the content” with “non-web documents or software”.

With this substitution, it would read:

**2.2.1 Timing Adjustable:** For each time limit that is set by **[non-web documents or software]**, at least one of the following is true: (Level A)

- **Turn off:** The user is allowed to turn off the time limit before encountering it; or
- **Adjust:** The user is allowed to adjust the time limit before encountering it over a wide range that is at least ten times the length of the default setting; or
- **Extend:** The user is warned before time expires and given at least 20 seconds to extend the time limit with a simple action (for example, “press the space bar”), and the user is allowed to extend the time limit at least ten times; or
- **Real-time Exception:** The time limit is a required part of a real-time event (for example, an auction), and no alternative to the time limit is possible; or
- **Essential Exception:** The time limit is [essential](#) and extending it would invalidate the activity; or
- **20 Hour Exception:** The time limit is longer than 20 hours.

#### NOTE

This success criterion helps ensure that users can complete tasks without unexpected changes in content or context that are a result of a time limit. This success criterion should be considered in conjunction with [Success Criterion 3.2.1](#), which puts limits on changes of content or context as a result of user action.

*Intent from Understanding Timing Adjustable*

The intent of this Success Criterion is to ensure that users with disabilities are given adequate time to interact with Web content whenever possible. People with disabilities such as blindness, low vision, dexterity impairments, and cognitive limitations may require more time to read content or to perform functions such as filling out on-line forms. If Web functions are time-dependent, it will be difficult for some users to perform the required action before a time limit occurs. This may render the service inaccessible to them. Designing functions that are not time-dependent will help people with disabilities succeed at completing these functions. Providing options to disable time limits, customize the length of time limits, or request more time before a time limit occurs helps those users who require more time than expected to successfully complete tasks. These options are listed in the order that will be most helpful for the user. Disabling time limits is better than customizing the length of time limits, which is better than requesting more time before a time limit occurs.

Any process that happens without user initiation after a set time or on a periodic basis is a time limit. This includes partial or full updates of content (for example, page refresh), changes to content, or the expiration of a window of opportunity for a user to react to a request for input.

It also includes content that is advancing or updating at a rate beyond the user's ability to read and/or understand it. In other words, animated, moving or scrolling content introduces a time limit on a users ability to read content.

This success criterion is generally not applicable when the content repeats or is synchronized with other content, so long as the information and data is adjustable or otherwise under the control of the end user. Examples of time limits for which this success criterion is not applicable include scrolling text that repeats, captioning, and [carousels](#). These are situations which do include time limits, but the content is still available to the user because it has controls for accessing it, as specified in [2.2.2 Pause, Stop, Hide](#).

In some cases, however, it is not possible to change the time limit (for example, for an auction or other real-time event) and exceptions are therefore provided for those cases.

### Notes regarding server time limits

- Timed server redirects can be found below under Common Failures.
- Non-timed server redirects (e.g., 3xx response codes) are not applicable because there is no time limit: they work instantly.

- This Success Criterion applies only to time limits that are set by the content itself. For example, if a time limit is included in order to address security concerns, it would be considered to have been set by the content because it is designed to be part of the presentation and interaction experience for that content. Time limits set externally to content, such as by the user agent or by factors intrinsic to the Internet are not under the author's control and not subject to WCAG conformance requirements. Time limits set by Web servers should be under the author's/organization's control and are covered. (Success Criteria [2.2.3](#), [2.2.4](#) and [2.2.5](#) may also apply.)
- Ten times the default was chosen based on clinical experience and other guidelines. For example, if 15 seconds is allowed for a user to respond and hit a switch, 150 seconds would be sufficient to allow almost all users to hit a switch even if they had trouble.
- 20 seconds was also based on clinical experience and other guidelines. 20 seconds to hit 'any switch' is sufficient for almost all users including those with spasticity. Some would fail, but some would fail all lengths of time. A reasonable period for requesting more time is required since an arbitrarily long time can provide security risks to all users, including those with disabilities, for some applications. For example, with kiosks or terminals that are used for financial transactions, it is quite common for people to walk away without signing off. This leaves them vulnerable to those walking up behind them. Providing a long period of inactivity before asking, and then providing a long period for the person to indicate that they are present can leave terminals open for abuse. If there is no activity the system should ask if the user is there. It should then ask for an indication that a person is there ('hit any key') and then wait long enough for almost anyone to respond. For "hit any key," 20 seconds would meet this. If the person indicates that they are still present, the device should return the user to the exact condition that existed before it asked the question.
- 20 hours was chosen as an upper limit because it is longer than a full waking day.

In cases where timing is not an intrinsic requirement but giving users control over timed events would invalidate the outcome, a third party can control the time limits for the user (for example, granting double time on a test).

See also [2.2.3: No Timing](#).

### § 6.2.3.3 Pause, Stop, Hide

For moving, blinking, scrolling, or auto-updating information, all of the following are true:

#### **Moving, blinking, scrolling**

For any moving, blinking or scrolling information that (1) starts automatically, (2) lasts more than five seconds, and (3) is presented in parallel with other content, there is a mechanism for the user to pause, stop, or hide it unless the movement, blinking, or scrolling is part of an activity where it is essential; and

#### **Auto-updating**

For any auto-updating information that (1) starts automatically and (2) is presented in parallel with other content, there is a mechanism for the user to pause, stop, or hide it or to control the frequency of the update unless the auto-updating is part of an activity where it is essential.

*For requirements related to flickering or flashing content, refer to [Guideline 2.3](#).*

*Since any content that does not meet this success criterion can interfere with a user's ability to use the whole page, all content on the Web page (whether it is used to meet other success criteria or not) must meet this success criterion. See [Conformance Requirement 5: Non-Interference](#).*

*Content that is updated periodically by software or that is streamed to the user agent is not required to preserve or present information that is generated or received between the initiation of the pause and resuming presentation, as this may not be technically possible, and in many situations could be misleading to do so.*

*An animation that occurs as part of a preload phase or similar situation can be considered essential if interaction cannot occur during that phase for all users and if not indicating progress could confuse users or cause them to think that content was frozen or broken.*

## § GUIDANCE WHEN APPLYING SUCCESS CRITERION 2.2.2 TO NON-WEB DOCUMENTS AND SOFTWARE

This applies directly as written, and as described in [Intent from Understanding Success Criterion 2.2.2](#) (also provided below), replacing “page” and “Web page” with “non-web documents and software” and removing “See Conformance Requirement 5: Non-Interference” in Note 2 of the success criterion.

With this substitution, it would read:

**2.2.2 Pause, Stop, Hide:** For moving, [blinking](#), scrolling, or auto-updating information, all of the following are true: (Level A)

- **Moving, blinking, scrolling:** For any moving, blinking or scrolling information that (1) starts automatically, (2) lasts more than five seconds, and (3) is presented in parallel with other content, there is a mechanism for the user to [pause](#), stop, or hide it unless the movement, blinking, or scrolling is part of an activity where it is [essential](#); and
- **Auto-updating:** For any auto-updating information that (1) starts automatically and (2) is presented in parallel with other content, there is a mechanism for the user to pause, stop, or hide it or to control the frequency of the update unless the auto-updating is part of an activity where it is essential.

#### NOTE 1

For requirements related to flickering or flashing content, refer to [Guideline 2.3](#).

#### NOTE 2

Since any [content](#) that does not meet this success criterion can interfere with a user's ability to use the whole **[non-web documents and software]**, all content on the **[non-web documents and software]** (whether it is used to meet other success criteria or not) must meet this success criterion.

#### NOTE 3

[Content](#) that is updated periodically by software or that is streamed to the user agent is not required to preserve or present information that is generated or received between the initiation of the pause and resuming presentation, as this may not be technically possible, and in many situations could be misleading to do so.

#### NOTE 4

An animation that occurs as part of a preload phase or similar situation can be considered essential if interaction cannot occur during that phase for all users and if not indicating progress could confuse users or cause them to think that content was frozen or broken.

#### NOTE 5

While the success criteria uses the term “information”, the WCAG 2.2 Intent section makes it clear that this is to be applied to all content. Any [content](#), whether informative or decorative, that is updated automatically, blinks, or moves may create an accessibility barrier.

*Intent from Understanding Pause, Stop, Hide*

The intent of this Success Criterion is to avoid distracting users during their interaction with a Web page.

"Moving, blinking and scrolling" refers to content in which the visible content conveys a sense of motion. Common examples include motion pictures, synchronized media presentations, animations, real-time games, and scrolling stock tickers. "Auto-updating" refers to content that updates or disappears based on a preset time interval. Common time-based content includes audio, automatically updated weather information, news, stock price updates, and auto-advancing presentations and messages. The requirements for moving, blinking and scrolling content and for auto-updating content are the same except that:

- authors have the option of providing the user with a means to control the frequency of updates when content is auto-updating and
- there is no five second exception for auto-updating since it makes little sense to auto-update for a few seconds and then stop

Content that moves or auto-updates can be a barrier to anyone who has trouble reading stationary text quickly as well as anyone who has trouble tracking moving objects. It can also cause problems for screen readers.

Moving content can also be a severe distraction for some people. Certain groups, particularly those with attention deficit disorders, find blinking content distracting, making it difficult for them to concentrate on other parts of the Web page. Five seconds was chosen because it is long enough to get a user's attention, but not so long that a user cannot wait out the distraction if necessary to use the page.

Content that is paused can either resume in real-time or continue playing from the point in the presentation where the user left off.

1. Pausing and resuming where the user left off is best for users who want to pause to read content and works best when the content is not associated with a real-time event or status.

*See [2.2.1: Timing Adjustable](#) for additional requirements related to time-limits for reading.*

2. Pausing and jumping to current display (when pause is released) is better for information that is real-time or "status" in nature. For example, weather radar, a stock ticker, a traffic camera, or an auction timer, would present misleading

information if a pause caused it to display old information when the content was restarted.

*Hiding content would have the same result as pausing and jumping to current display (when pause is released).*

For a mechanism to be considered "a mechanism for the user to pause," it must provide the user with a means to pause that does not tie up the user or the focus so that the page cannot be used. The word "pause" here is meant in the sense of a "pause button" although other mechanisms than a button can be used. Having an animation stop only so long as a user has focus on it (where it restarts as soon as the user moves the focus away) would not be considered a "mechanism for the user to pause" because it makes the page unusable in the process and would not meet this SC.

It is important to note that the terms "blinking" and "flashing" can sometimes refer to the same content.

- "Blinking" refers to content that causes a distraction problem. Blinking can be allowed for a short time as long as it stops (or can be stopped)
- "Flashing" refers to content that can trigger a seizure (if it is more than 3 per second and large and bright enough). This cannot be allowed even for a second or it could cause a seizure. And turning the flash off is also not an option since the seizure could occur faster than most users could turn it off.
- Blinking usually does not occur at speeds of 3 per second or more, but it can. If blinking occurs faster than 3 per second, it would also be considered a flash.

## 6.2.4 Seizures and Physical Reactions

Do not design content in a way that is known to cause seizures or physical reactions.

### Guidance When Applying Guideline 2.3 to Non-Web Documents and Software

In WCAG 2.2, the Guidelines are provided for framing and understanding the success

criteria under them but are not required for conformance to WCAG. Guideline 2.3 applies directly as written.

### *Intent from Understanding Seizures and Physical Reactions*

Some people with seizure disorders can have a seizure triggered by flashing visual content. Most people are unaware that they have this disorder until it strikes. In 1997, a cartoon on television in Japan sent over 700 children to the hospital, including about 500 who had seizures. Warnings do not work well because they are often missed, especially by children who may in fact not be able to read them.

The objective of this guideline is to ensure that content that is marked as conforming to WCAG 2.0 avoids the types of flash that are most likely to cause seizure when viewed even for a second or two.

#### § 6.2.4.2 *Three Flashes or Below Threshold*

Web pages do not contain anything that flashes more than three times in any one second period, or the flash is below the general flash and red flash thresholds.

*Since any content that does not meet this success criterion can interfere with a user's ability to use the whole page, all content on the Web page (whether it is used to meet other success criteria or not) must meet this success criterion. See [Conformance Requirement 5: Non-Interference](#).*

### *Intent from Understanding Three Flashes or Below Threshold*

The intent of this Success Criterion is to allow users to access the full content of a site without inducing seizures due to photosensitivity.

Individuals who have photosensitive seizure disorders can have a seizure triggered by content that flashes at certain frequencies for more than a few flashes. People are even more sensitive to red flashing than to other colors, so a special test is provided for saturated red flashing. These guidelines were originally based on guidelines for the broadcasting industry as adapted for desktop monitors, where content is viewed from a closer distance (using a larger angle of vision).

Flashing can be caused by the display, the computer rendering the image or by the content being rendered. The author has no control of the first two. They can be addressed by the design and speed of the display and computer. The intent of this criterion is to ensure that flicker that violates the flash thresholds is not caused by the content itself. For example, the content could contain a video clip or animated image of a series of strobe flashes, or close-ups of rapid-fire explosions.

This Success Criterion replaces a much more restrictive criterion in WCAG 1.0 that did not allow any flashing (even of a single pixel) within a broad frequency range (3 to 50 Hz). This Success Criterion is based on existing specifications in use in the UK and by others for television broadcast and has been adapted for computer display viewing. In WCAG 2.0, a 1024 x 768 screen was used as the reference screen resolution for the evaluation. The 341 x 256 pixel block represents a 10 degree viewport at a typical viewing distance. (The 10 degree field is taken from the original specifications and represents the central vision portion of the eye, where people are most susceptible to photo stimuli.)

With the proliferation of devices of varying screen sizes (from small hand-helds to large living room displays), as well as the adoption of [CSS pixels](#) as a density-independent unit of measurement, the prior assessment criteria may seem outdated. However, an image of a consistent size uses up relatively the same percentage of a user's visual field on any device. On a large screen, the image takes up less size, but the large screen takes up a larger part of the visual field. On a mobile screen, the image may take up most or all of the screen; however, the mobile screen itself takes up a smaller portion of the user's visual field. So the same dimension of the flashing content, represented in CSS pixels can still provide a consistent means of assessment. Substituting CSS pixels for the original pixel block means that the combined area of flashing becomes 341 x 256 CSS pixels, or a flashing area of 87,296 CSS pixels.

Content should be analyzed at the largest scale at which a user may view the content,

and at the standard zoom level of the user agent. For example, with a video that may play in an area of a web page and also at full screen, the video should be analyzed for risks at full screen.

Where video content is provided in color spaces other than sRGB, the version provided with the highest dynamic range should be tested. In such cases the industry standard definition of a general flash is a change in luminance of 20 cd/m<sup>2</sup> or more where the darker image is below 160 cd/m<sup>2</sup>. ([ITU-R BT.1702](#).) This is applicable for standard dynamic range (SDR) and high dynamic range (HDR) content. For HDR content when the darker state is 160 cd/m<sup>2</sup> or more, a general flash is one where the Michelson contrast is 1/17 or greater — where the Michelson contrast is calculated as  $(L_{High} - L_{Low}) / (L_{High} + L_{Low})$ , and where  $L_{High}$  and  $L_{Low}$  are the luminance of the high and low luminance states, respectively.

For short clips that might be looped (such as GIF animations), the content should be analyzed while looping.

*The specification cannot account for the actual viewing distance that a person chooses. Users that are closer to their screens than the idealized viewing distance will be affected by flashing areas that normatively pass. The same problem applies to users who rely on zoom or screen magnification. Conversely, users who are further away from the screen than the idealized distance should be able to tolerate flashing areas that are larger than the threshold.*

The combined area of flashes occurring concurrently and contiguously means the total area that is actually flashing at the same time. It is calculated by adding up the contiguous area that is flashing simultaneously within any 10 degree angle of view.

*The terms "blinking" and "flashing" can sometimes refer to the same content.*

- *"Blinking" refers to content that causes a distraction problem. Blinking can be allowed for a short time as long as it stops (or can be stopped)*
- *"Flashing" refers to content that can trigger a seizure (if it is more than 3 per second and large and bright enough). This cannot be allowed even for a second or it could cause a seizure. And turning the flash off is also not an option since the seizure could occur faster than most users could turn it off.*
- *Blinking usually does not occur at speeds of 3 per second or more, but it can. If blinking occurs faster than 3 per second, it would also be considered a flash.*

The new (in WCAG 2.2) working definition in the field for "**pair of opposing transitions involving a saturated red**" is a pair of opposing transitions where, one transition is either to or from a state with a value  $R/(R + G + B)$  that is greater than or equal to 0.8, and the difference between states is more than 0.2 (unitless) in the CIE 1976 UCS chromaticity diagram. [ISO 9241-391]

The chromaticity difference is calculated as:

$$\bullet \text{ SQRT}( (u'1 - u'2)^2 + (v'1 - v'2)^2 )$$

where  $u'1$  and  $v'1$  are chromaticity coordinates of State 1 and  $u'2$  and  $v'2$  are chromaticity coordinates of State 2. The 1976 UCS chromaticity coordinates of  $u'$  and  $v'$  are calculated as:

$$\bullet u' = 4 * X / (X + 15 * Y + 3 * Z)$$

$$\bullet v' = 9 * Y / (X + 15 * Y + 3 * Z)$$

where  $X$ ,  $Y$ , and  $Z$  are the tristimulus values of a color in the CIE XYZ colorspace, which can be calculated as:

$$\bullet X = 0.4124564 * R + 0.3575761 * G + 0.1804375 * B$$

$$\bullet Y = 0.2126729 * R + 0.7151522 * G + 0.0721750 * B$$

$$\bullet Z = 0.0193339 * R + 0.1191920 * G + 0.9503041 * B$$

where  $R$ ,  $G$ , &  $B$  are values that range from 0-1 as specified in "relative luminance" definition.

## § Guidance When Applying Success Criterion 2.3.1 to Non-Web Documents and Software

This applies directly as written, and as described in [Intent from Understanding Success Criterion 2.3.1](#) (also provided below), replacing "Web pages" with "non-web documents or software", "the whole page" with "the whole non-web document or software", "the Web page" with "the non-web document or software", and removing "See Conformance Requirement 5: Non-Interference".

With these substitutions, it would read:

**2.3.1 Three Flashes or Below Threshold:** **[Non-web documents or software]** do not contain anything that flashes more than three times in any one second period, or the [flash](#) is below the [general flash and red flash thresholds](#). (Level A)

#### NOTE

Since any content that does not meet this success criterion can interfere with a user's ability to use the whole **[non-web document or software]**, all content on the **[non-web document or software]** (whether it is used to meet other success criteria or not) must meet this success criterion.

## § 6.2.5 Navigable

Provide ways to help users navigate, find content, and determine where they are.

### § *Guidance When Applying Guideline 2.4 to Non-Web Documents and Software*

In WCAG 2.2, the Guidelines are provided for framing and understanding the success criteria under them but are not required for conformance to WCAG. Guideline 2.4 applies directly as written.

*Intent from Understanding Navigable*

The intent of this guideline is to help users find the content they need and allow them to keep track of their location. These tasks are often more difficult for people with disabilities. For finding, navigation, and orientation, it is important that the user can find out what the current location is. For navigation, information about the possible destinations needs to be available. Screen readers convert content to synthetic speech which, because it is audio, must be presented in linear order. Some Success Criteria in this guideline explain what provisions need to be taken to ensure that screen reader users can successfully navigate the content. Others allow users to more easily recognize navigation bars and page headers and to bypass this repeated content. Unusual user interface features or behaviors may confuse people with cognitive disabilities.

Navigation has two main functions:

- to tell the user where they are
- to enable the user to go somewhere else

This guideline works closely with [Guideline 1.3](#), which ensures that any structure in the content can be perceived, a key to navigation as well. Headings are particularly important mechanisms for helping users orient themselves within content and navigate through it. Many users of assistive technologies rely on appropriate headings to skim through information and easily locate the different sections of content. Satisfying [Success Criterion 1.3.1](#) for headings also addresses some aspects of Guideline 2.4.

#### § 6.2.5.2 Bypass Blocks

A mechanism is available to bypass blocks of content that are repeated on multiple Web pages.

#### § GUIDANCE WHEN APPLYING SUCCESS CRITERION 2.4.1 TO NON-WEB DOCUMENTS AND SOFTWARE

This applies directly as written and described in [Intent from Understanding Success](#)

[Criterion 2.4.1](#) (also provided below), replacing “Web pages” with “non-web documents in a set of non-web documents” or “software programs in a set of software programs” to explicitly state that the multiple documents (or software programs) are part of a set rather than any two documents or pieces of software.

With these substitutions, this success criterion would read:

(for non-web documents)

**2.4.1 Bypass Blocks:** A [mechanism](#) is available to bypass blocks of content that are repeated on multiple [\[non-web documents in a set of non-web documents\]](#).

(for software programs)

**2.4.1 Bypass Blocks:** A mechanism is available to bypass blocks of content that are repeated on multiple [\[software programs in a set of software programs\]](#).

#### NOTE 1

See [set of documents](#) and [set of software programs](#) in the Key Terms section of the Introduction to determine when a group of documents or pieces of software is considered a set for this success criterion. (Sets of software that meet this definition appear to be extremely rare.)

#### NOTE 2

Individual documents or software programs (not in a set) would automatically meet this success criterion because this success criterion applies only to things that appear in a set.

#### NOTE 3

Although not required by the success criterion, being able to bypass blocks of content that are repeated *within* non-web documents or software directly addresses user needs identified in the Intent section for this Success Criterion, and is generally considered best practice.

NOTE 4

Many software user interface components have built-in mechanisms to navigate directly to / among them, which also have the effect of skipping over or bypassing blocks of content.

*Intent from Understanding Bypass Blocks*

The intent of this Success Criterion is to allow people who navigate sequentially through content more direct access to the primary content of the Web page. Web pages and applications often have content that appears on other pages or screens. Examples of repeated blocks of content include but are not limited to navigation links, heading graphics, and advertising frames. Small repeated sections such as individual words, phrases or single links are not considered blocks for the purposes of this provision.

This is in contrast to a sighted user's ability to ignore the repeated material either by focusing on the center of the screen (where main content usually appears) or a mouse user's ability to select a link with a single mouse click rather than encountering every link or form control that comes before the item they want.

It is not the intent of this Success Criterion to require authors to provide methods that are redundant to functionality provided by the user agent. Most web browsers provide keyboard shortcuts to move the user focus to the top of the page, so if a set of navigation links is provided at the bottom of a web page providing a "skip" link may be unnecessary.

*Although this Success Criterion deals with blocks of content that are repeated on multiple pages, we also strongly promote structural markup on individual pages as per Success Criteria 1.3.1.*

Although the success criterion does not specifically use the term “within a set of web pages”, the concept of the pages belonging to a set is implied. An author would not be expected to avoid any possible duplication of content in any two pages that are not in some way related to each other; that are not "Web pages that share a common purpose and that are created by the same author, group or organization" (the definition of set of web pages).

*Even for web pages that are not in a set, if a web page has blocks of text that are repeated within the page it may be helpful (but not required) to provide a means to skip over them.*

### § 6.2.5.3 Page Titled

Web pages have titles that describe topic or purpose.

## § GUIDANCE WHEN APPLYING SUCCESS CRITERION 2.4.2 TO NON-WEB DOCUMENTS AND SOFTWARE

This applies directly as written, and as described in [Intent from Understanding Success Criterion 2.4.2](#) (also provided below) replacing “Web pages” with “non-web documents or software”.

With this substitution, it would read:

**2.4.2 Page Titled:** **[Non-web documents or software]** have titles that describe topic or purpose. (Level A)

### NOTE 1

As described in the WCAG intent (also provided below), the name of a [non-web software application](#) or [non-web document](#) (e.g. document, media file, etc.) is a sufficient title if it describes the topic or purpose.

### NOTE 2

See also the discussion on [Closed Functionality](#).

*Intent from Understanding Page Titled*

The intent of this Success Criterion is to help users find content and orient themselves within it by ensuring that each Web page has a descriptive title. Titles identify the current location without requiring users to read or interpret page content. When titles appear in site maps or lists of search results, users can more quickly identify the content they need. User agents make the title of the page easily available to the user for identifying the page. For instance, a user agent may display the page title in the window title bar or as the name of the tab containing the page.

In cases where the page is a document or a web application, the name of the document or web application would be sufficient to describe the purpose of the page. Note that it is not required to use the name of the document or web application; other things may also describe the purpose or the topic of the page.

[Success Criteria 2.4.4](#) and [2.4.9](#) deal with the purpose of links, many of which are links to web pages. Here also, the name of a document or web application being linked to would be sufficient to describe the purpose of the link. Having the link and the title agree, or be very similar, is good practice and provides continuity between the link 'clicked on' and the web page that the user lands on.

#### § 6.2.5.4 Focus Order

If a Web page can be navigated sequentially and the navigation sequences affect meaning or operation, focusable components receive focus in an order that preserves meaning and operability.

#### § GUIDANCE WHEN APPLYING SUCCESS CRITERION 2.4.3 TO NON-WEB DOCUMENTS AND SOFTWARE

This applies directly as written, and as described in [Intent from Understanding Success Criterion 2.4.3](#) (also provided below) replacing “a Web page” with “non-web documents or software”.

With this substitution, it would read:

**2.4.3 Focus Order:** If [\[non-web documents or software\]](#) can be navigated sequentially and

the navigation sequences affect meaning or operation, focusable components receive focus in an order that preserves meaning and operability. (Level A)

*Intent from Understanding Focus Order*

The intent of this Success Criterion is to ensure that when users navigate sequentially through content, they encounter information in an order that is consistent with the meaning of the content and can be operated from the keyboard. This reduces confusion by letting users form a consistent mental model of the content. There may be different orders that reflect logical relationships in the content. For example, moving through components in a table one row at a time or one column at a time both reflect the logical relationships in the content. Either order may satisfy this Success Criterion.

The way that sequential navigation order is determined in Web content is defined by the technology of the content. For example, simple HTML defines sequential navigation via the notion of tabbing order. Dynamic HTML may modify the navigation sequence using scripting along with the addition of a `tabindex` attribute to allow focus to additional elements. If no scripting or `tabindex` attributes are used, the navigation order is the order that components appear in the content stream. (See HTML 4.01 Specification, section 17.11, "Giving focus to an element").

An example of keyboard navigation that is not the sequential navigation addressed by this Success Criterion is using arrow key navigation to traverse a tree component. The user can use the up and down arrow keys to move from tree node to tree node. Pressing the right arrow key may expand a node, then using the down arrow key, will move into the newly expanded nodes. This navigation sequence follows the expected sequence for a tree control - as additional items get expanded or collapsed, they are added or removed from the navigation sequence.

The focus order may not be identical to the programmatically determined reading order (see Success Criterion 1.3.2) as long as the user can still understand and operate the Web page. Since there may be several possible logical reading orders for the content, the focus order may match any of them. However, when the order of a particular presentation differs from the programmatically determined reading order, users of one of these presentations may find it difficult to understand or operate the Web page. Authors should carefully consider all these users as they design their Web pages.

For example, a screen reader user interacts with the programmatically determined reading order, while a sighted keyboard user interacts with the visual presentation of the Web page. Care should be taken so that the focus order makes sense to both of these sets of users and does not appear to either of them to jump around randomly.

For clarity:

1. Focusable components need to receive focus in an order that preserves meaning and operability only when navigation sequences affect meaning and operability.
2. In those cases where it is required, there may be more than one order that will preserve meaning and operability.
3. If there is more than one order that preserves meaning and operability, only one of them needs to be provided.

#### § 6.2.5.5 *Link Purpose (In Context)*

The purpose of each link can be determined from the link text alone or from the link text together with its programmatically determined link context, except where the purpose of the link would be ambiguous to users in general.

#### § GUIDANCE WHEN APPLYING SUCCESS CRITERION 2.4.4 TO NON-WEB DOCUMENTS AND SOFTWARE

This applies directly as written and as described in [Intent from Understanding Success Criterion 2.4.4](#) (also provided below).

##### NOTE

In software, a “link” is any text string or image in the user interface outside a user interface control that behaves like a hypertext link. This does not include general user interface controls or buttons. (An OK button, for example, would not be a link.)

#### *Intent from Understanding Link Purpose (In Context)*

The intent of this Success Criterion is to help users understand the purpose of each link so they can decide whether they want to follow the link. Whenever possible, provide link text that identifies the purpose of the link without needing additional context. Assistive technology has the ability to provide users with a list of links that are on the Web page. Link text that is as meaningful as possible will aid users who want to choose from this list of links. Meaningful link text also helps those who wish to tab from link to link. Meaningful links help users choose which links to follow without requiring complicated strategies to understand the page.

The text of, or associated with, the link is intended to describe the purpose of the link. In cases where the link takes one to a document or a web application, the name of the document or web application would be sufficient to describe the purpose of the link (which is to take you to the document or web application). Note that it is not required to use the name of the document or web application; other things may also describe the purpose of the link.

[Success Criterion 2.4.2](#) deals with the titles of pages. Here also, the name of a document or web application being presented on the page would be sufficient to describe the purpose of the page. Having the link and the title agree, or be very similar, is good practice and provides continuity between the link 'clicked on' and the web page that the user lands on.

In some situations, authors may want to provide part of the description of the link in logically related text that provides the context for the link. In this case the user should be able to identify the purpose of the link without moving focus from the link. In other words, they can arrive on a link and find out more about it without losing their place. This can be achieved by putting the description of the link in the same sentence, paragraph, list item, or table cell as the link, or in the table header cell for a link in a data table, because these are directly associated with the link itself. Alternatively, authors may choose to use an ARIA technique to associate additional text on the page with the link.

This context will be most usable if it precedes the link. (For instance, if you must use ambiguous link text, it is better to put it at the end of the sentence that describes its destination, rather than putting the ambiguous phrase at the beginning of the sentence.) If the description follows the link, there can be confusion and difficulty for screen reader users who are reading through the page in order (top to bottom).

It is a best practice for links with the same destination to have consistent text (and this is a requirement per [Success Criterion 3.2.4](#) for pages in a set). It is also a best practice

for links with different purposes and destinations to have different link text.

A best practice for links to conforming alternate versions is to ensure that the link text to the conforming alternate version indicates in link text that the page it leads to represents the more accessible version. This information may also be provided in text - the goal is to ensure that the end user knows what the purpose of the link is.

The Success Criterion includes an exception for links for which the purpose of the link cannot be determined from the information on the Web page. In this situation, the person with the disability is not at a disadvantage; there is no additional context available to understand the link purpose. However, whatever amount of context is available on the Web page that can be used to interpret the purpose of the link must be made available in the link text or programmatically associated with the link to satisfy the Success Criterion.

*There may be situations where the purpose of the link is supposed to be unknown or obscured. For instance, a game may have links identified only as door #1, door #2, and door #3. This link text would be sufficient because the purpose of the links is to create suspense for all users.*

See also [2.4.9: Link Purpose \(Link Only\)](#).

#### § 6.2.5.6 Multiple Ways

More than one way is available to locate a Web page within a set of Web pages except where the Web Page is the result of, or a step in, a process.

#### § GUIDANCE WHEN APPLYING SUCCESS CRITERION 2.4.5 TO NON-WEB DOCUMENTS AND SOFTWARE

This applies directly as written and described in [Intent from Understanding Success Criterion 2.4.5](#) (also provided below), replacing “set of Web pages” with “set of non-web documents” and “set of software programs”.

With these substitutions, this success criterion would read:

(for non-web documents)

**2.4.5 Multiple Ways:** More than one way is available to locate a [\[non-web document\]](#) within a [\[set of non-web documents\]](#) except where the [\[non-web document\]](#) is the result of, or a step in, a [process](#).

(for software programs)

**2.4.5 Multiple Ways:** More than one way is available to locate a [\[software program\]](#) within a [\[set of software programs\]](#) except where the [\[software program\]](#) is the result of, or a step in, a [process](#).

#### NOTE 1

See [set of documents](#) and [set of software programs](#) in the Key Terms section of the Introduction to determine when a group of documents or software is considered a set for this success criterion. (Sets of software that meet this definition appear to be extremely rare.)

#### NOTE 2

The definitions of “[set of documents](#)” and “[set of software programs](#)” in the Key Terms section of the Introduction are predicated on the ability to navigate from each element of the set to each other, and navigation is a type of locating. So the mechanism used to navigate between elements of the set will be one way of locating information in the set. Non-web environments, generally major operating systems with browse and search capabilities, often provide infrastructure and tools that provide mechanisms for locating content in a set of non-web documents or a set of software programs. For example, it may be possible to browse through the files or programs that make up a set, or search within members of the set for the names of other members. A file directory would be the equivalent of a site map for documents in a set, and a search function in a file system would be equivalent to a web search function for web pages. Such facilities may provide additional ways of locating information in the set.

### NOTE 3

An example of the use of “a software program that is part of process”, that would meet the exception for this Success Criterion, would be one where programs are interlinked but the interlinking depends on program A being used before program B, for validation or to initialize the dataset etc.

### NOTE 4

While some users may find it useful to have multiple ways to locate some groups of user interface elements within a document or software program, this is not required by the success criterion (and may pose difficulties in some situations).

### NOTE 5

The definitions of “[set of documents](#)” and “[set of software programs](#)” in WCAG2ICT require every item in the set to be independently reachable, and so nothing in such a set can be a “step in a process” that can't be reached any other way. The purpose of the exception—that items in a process are exempt from meeting this success criterion—is achieved by the definition of set.

## *Intent from Understanding Multiple Ways*

The intent of this Success Criterion is to make it possible for users to locate content in a manner that best meets their needs. Users may find one technique easier or more comprehensible to use than another.

Even small sites should provide users some means of orientation. For a three or four page site, with all pages linked from the home page, it may be sufficient simply to provide links from and to the home page where the links on the home page can also serve as a site map.

## § 6.2.5.7 Headings and Labels

Headings and labels describe topic or purpose.

## § GUIDANCE WHEN APPLYING SUCCESS CRITERION 2.4.6 TO NON-WEB DOCUMENTS AND SOFTWARE

This applies directly as written, and as described in [Intent from Understanding Success Criterion 2.4.6](#) (also provided below).

### NOTE

In [software](#), headings and labels are used to describe sections of [content](#) and controls respectively. In some cases it may be unclear whether a piece of static text is a heading or a label. But whether treated as a label or a heading, the requirement is the same: that if they are present they describe the topic or purpose of the item(s) they are associated with.

*Intent from Understanding Headings and Labels*

The intent of this Success Criterion is to help users understand what information is contained in Web pages and how that information is organized. When headings are clear and descriptive, users can find the information they seek more easily, and they can understand the relationships between different parts of the content more easily. Descriptive labels help users identify specific components within the content.

Labels and headings do not need to be lengthy. A word, or even a single character, may suffice if it provides an appropriate cue to finding and navigating content.

This Success Criterion does not require headings or labels. This Success Criterion requires that if headings or labels are provided, they be descriptive. This Success Criterion also does not require that content acting as a heading or label be correctly marked up or identified - this aspect is covered separately by [1.3.1: Info and Relationships](#). It is possible for content to pass this Success Criterion (providing descriptive content that acts as headings or labels) while failing Success Criterion 1.3.1 (if the headings or labels aren't correctly marked up/identified). Conversely, it is also possible for content to pass Success Criterion 1.3.1 (with headings or labels correctly marked up or identified), while failing this Success Criterion (if those headings or labels are not sufficiently clear or descriptive).

Further, in the case of labels, this Success Criterion does not take into consideration whether or not alternative methods of providing an accessible name for form controls and inputs has been used - this aspect is covered separately by [4.1.2: Name, Role and Value](#). It is possible for controls and inputs to have an appropriate accessible name (e.g. using `aria-label=" . . . "`) and therefore pass Success Criterion 4.1.2, but to still fail this Success Criterion (if the label is not sufficiently clear or descriptive).

This success criterion does not require the use of labels; however, it does require that if labels are present, they must be sufficiently clear or descriptive. Please see [3.3.2: Labels or Instructions](#) for more information on the use of labels.

#### § 6.2.5.8 Focus Visible

Any keyboard operable user interface has a mode of operation where the keyboard focus indicator is visible.

## § GUIDANCE WHEN APPLYING SUCCESS CRITERION 2.4.7 TO NON-WEB DOCUMENTS AND SOFTWARE

This applies directly as written, and as described in [Intent from Understanding Success Criterion 2.4.7](#) (also provided below).

### *Intent from Understanding Focus Visible*

The purpose of this success criterion is to help a person know which element has the keyboard focus.

“Mode of operation” accounts for user agents which may not always show a focus indicator, or only show the focus indicator when the keyboard is used. User agents may optimise when the focus indicator is shown, such as only showing it when a keyboard is used. Authors are responsible for providing at least one mode of operation where the focus is visible. In most cases there is only one mode of operation so this success criterion applies. The focus indicator must not be time limited, when the keyboard focus is shown it must remain.

Note that a keyboard focus indicator can take different forms. This criterion does not specify what the form is, but [Focus Appearance](#) does define how visible the indicator should be. Passing [Focus Appearance \(Minimum\)](#) would pass this success criterion.

### § 6.2.5.9 Focus Not Obscured (Minimum)

When a user interface component receives keyboard focus, the component is not entirely hidden due to author-created content.

*If the interface is configurable so that the user can reposition content such as toolbars and non-modal dialogs, then only the initial positions of user-movable content are considered for testing and conformance of this Success Criterion.*

## § GUIDANCE WHEN APPLYING SUCCESS CRITERION 2.4.11 TO NON-WEB DOCUMENTS AND SOFTWARE

### EDITOR'S NOTE

This section is to be developed by the WCAG2ICT Task Force.

*Intent from Understanding Focus Not Obscured (Minimum)*

The intent of this Success Criterion is to ensure that the item receiving keyboard focus is always partially visible in the user's viewport. For sighted people who rely on a keyboard (or on a device that operates through the keyboard interface, such as a switch or voice input), knowing the current point of focus is critical. The component with focus signals the interaction point on the page. Where users cannot see the item with focus, they may not know how to proceed, or may even think the system has become unresponsive.

In recognition of the complex responsive designs common today, this AA criterion allows for the component receiving focus to be *partially* obscured by other author-created content. A partly obscured component can still be very visible, although the more of it that is obscured, the less easy it is to see. For that reason, authors should attempt to design interactions to reduce the degree and frequency with which the item receiving focus is partly obscured. For best visibility, *none* of the component receiving focus should be hidden. This preferred outcome is covered by the AAA criterion [Focus Not Obscured \(Enhanced\)](#).

Typical types of content that can overlap focused items are sticky footers, sticky headers, and non-modal dialogs. As a user tabs through the page, these layers of content can obscure the item receiving focus, along with its focus indicator.

A notification implemented as sticky content, such as a cookie banner, will fail this Success Criterion if it entirely obscures a component receiving focus. Ways of passing include making the banner modal so the user has to dismiss the banner before navigating through the page, or using [scroll padding](#) so the banner does not overlap other content. Notifications that do not require user action could also meet this criterion by closing on loss of focus.

Another form of obscuring can occur where light boxes or other semi-opaque effects overlap the item with focus. While less than 100 percent opacity is not causing the component to be “entirely hidden”, such semi-opaque overlaps may cause a failure of [1.4.11 Non-text Contrast](#). When a focus indicator can be covered by a semi-opaque component, the ability of the focus indicator to pass 1.4.11 should be evaluated (and pass) while the focus indicator is under the semi-opaque component. The intention in both situations is that the component receiving focus should never be obscured to the point a user cannot tell which item has focus.

## § 6.2.6 Input Modalities

Make it easier for users to operate functionality through various inputs beyond keyboard.

## § *Guidance When Applying Guideline 2.5 to Non-Web Documents and Software*

In WCAG 2.2, the Guidelines are provided for framing and understanding the success criteria under them but are not required for conformance to WCAG. Guideline 2.5 applies directly as written.

### *Intent from Understanding Input Modalities*

All functionality should be accessible via pointer input devices, for example, via a mouse pointer, a finger interacting with a touch screen, an electronic pencil/stylus, or a laser pointer.

People operating pointer input devices may not be able to carry out timed or complex gestures. Examples are drag-and-drop gestures and on touch screens, swiping gestures, split taps, or long presses. This Guideline does not discourage the provision of complex and timed gestures by authors. However, where they are used, an alternative method of input should be provided to enable users with motor impairments to interact with content via single untimed pointer gestures.

Often, people use devices that offer several input methods, for example, mouse input, touch input, keyboard input, and speech input. These should be supported concurrently as users may at any time switch preferred input methods due to situational circumstances, for example, the availability of a flat support for mouse operation, or situational impediments through motion or changes of ambient light.

A common requirement for pointer interaction is the ability of users to position the pointer over the target. With touch input, the pointer (the finger) is larger and less precise than a mouse cursor. For people with motor impairments, a larger target makes it easier to successfully position the pointer and activate the target.

### § 6.2.6.2 Pointer Gestures

All functionality that uses multipoint or path-based gestures for operation can be operated with a single pointer without a path-based gesture, unless a multipoint or path-based gesture is essential.

*This requirement applies to web content that interprets pointer actions (i.e. this does not apply to actions that are required to operate the user agent or assistive technology).*

## § GUIDANCE WHEN APPLYING SUCCESS CRITERION 2.5.1 TO NON-WEB DOCUMENTS AND SOFTWARE

### EDITOR'S NOTE

The WCAG2ICT Task Force seeks input on whether there are other examples of non-web documents that support the ability for authors to add gesture actions to the document that are not interpreted and acted upon through a user agent. If such example exist, we need broaden our example beyond prototyping software.

This applies directly as written, and as described in [Intent from Understanding Success Criterion 2.5.1](#) (also provided below), making changes to the notes for non-web documents by replacing “web content” with “content” and for non-web software by replacing “web content” with “non-web software” and “user agent” with “underlying platform software”.

With these substitutions, the notes would read:

(non-web documents)

### NOTE 1

This requirement applies to **[content]** that interprets pointer actions (i.e. this does not apply to actions that are required to operate the user agent or assistive technology).

#### NOTE 2

Multipoint and path-based gestures are less common in documents. An example where a document author could add such gestures is an interactive prototype document created in a software design tool.

(non-web software)

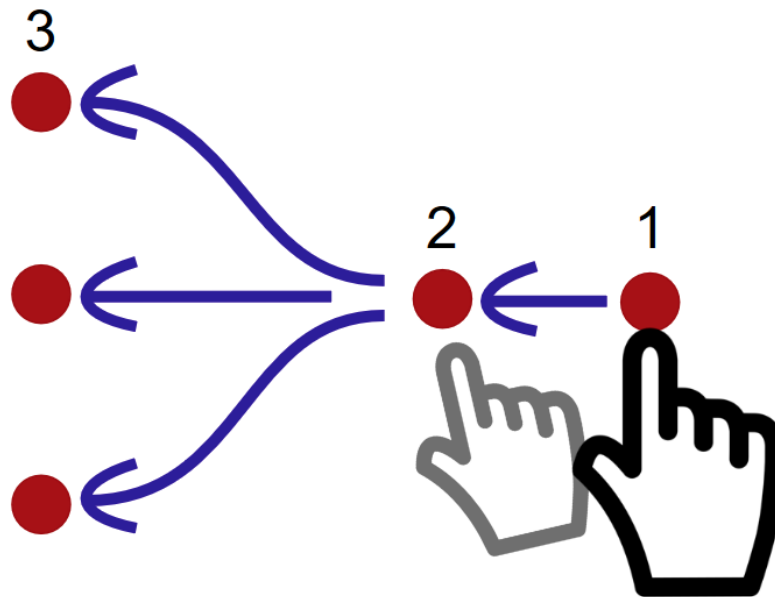
#### NOTE 3

This requirement applies to **[non-web software]** that interprets pointer actions (i.e. this does not apply to actions that are required to operate the **[underlying platform software]** or assistive technology).

*Intent from Understanding Pointer Gestures*

The intent of this Success Criterion is to ensure that content can be controlled with a range of pointing devices, abilities, and assistive technologies. Some people cannot perform gestures in a precise manner, or they may use a specialized or adapted input device such as a head pointer, eye-gaze system, or speech-controlled mouse emulator. Some pointing methods lack the capability or accuracy to perform multipoint or path-based gestures.

A **path-based gesture** involves an interaction where not just the endpoints matter. If going through an intermediate point (usually near the start of the gesture) also affects its meaning then it is a path-based gesture. The user engages a pointer (starting point), carries out a movement that goes through at least one intermediate-point before disengaging the pointer (end point). The intermediate point defines the gesture as requiring a specific path, even if the complete path is not defined.

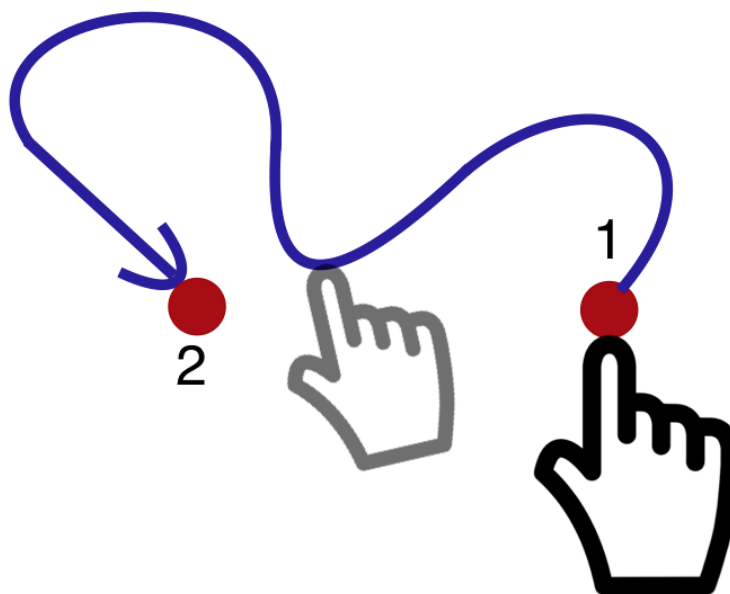


*A path-based gesture involves starting a pointer movement that goes through at least one intermediate point before the end-point. The end-point may be a continuation, or allow for various movements.*

Examples of path-based gestures include swiping, sliders and carousels dependent on the direction of interaction, and other gestures which trace a prescribed path such as drawing a specific shape. Such paths may be drawn with a finger or stylus on a touchscreen, graphics tablet, or trackpad, or with a mouse, joystick, or similar pointer device.

Dragging is a movement where the user picks up an object with a pointer (such as

mouse cursor or a finger) and moves it to some other position. This movement from start point to end point does not require the user to follow any particular path or direction. Dragging is therefore not path-based. In contrast, a path-based pointer gesture requires the traversal of an intermediate point, which is a technical way of expressing that the directionality and possibly speed of the gesture communicates a particular command to the system. Dragging motions are covered in [Success Criterion 2.5.7: Dragging](#).



*A free-form gesture does not require any particular path before the end-point, only the start and (optionally) the end point matter.*

*Any movement of a pointer could be difficult or impossible to use for someone who cannot perform precise movements, therefore alternative forms of interaction are always recommended. This success criterion is scoped to path-based gestures as it may be difficult or impossible to provide an alternative for free-form paths.*

Examples of **multipoint** gestures include a two-finger pinch zoom, a split tap where one finger rests on the screen and a second finger taps, or a two- or three-finger tap or swipe. Users may find it difficult or impossible to accomplish these if they type and point with a single finger or stick.

Authors must ensure that their content can be operated without multipoint or path-based gestures. Multipoint or path-based gestures can be used so long as the functionality can also be operated by another method, such as a tap, click, double tap,

double click, long press, or click & hold.

This Success Criterion applies to gestures in the author-provided content, not gestures defined by the operating system, user agent, or assistive technology. Examples of operating system gestures would be swiping down to see system notifications and gestures for built-in assistive technologies (AT). Examples of user agent-implemented gestures would be horizontal swiping implemented by browsers for navigating within the page history, or vertical swiping to scroll page content.

There are times when a component requires a path-based gesture for touch screen devices but not with a mouse. Taking an example of a generic slider:

- **Using a mouse:** If the user clicks on the thumb control of the slider and moves vertically, the slider will respond by moving to the right or left, even if the movement is mostly upwards. There will be no page scrolling as a result of the vertical movement as long as they drag with focus on the slider. Therefore, the slider does not require a path-based gesture with mouse pointer.
- **Using a touch-screen:** If the user puts their finger on the thumb control of the slider and moves upwards more than sideways, the slider may not respond because the browser takes control of the swipe and interprets it as a scroll, and will move the page up and down. Moving left or right on the slider thumb engages the slider and then the user can vary their vertical movement. This implementation has the 3-point requirement to work with a finger on a touch screen device so is a path-based gesture.

As touch screen devices can apply default gestures it is important to test with them if you are unsure whether a particular component does require a path-based gesture.

Browsers on a touch screen device generally provide some default gestures that impact whether a path-based gesture is needed. For example, a web browser on a touch-screen devices might detect a vertical gesture and scroll the page. If a user places their finger on a slider thumb and moves up (to scroll down) that might not activate the slider (depending on implementation). If the user moves horizontally first then the slider could capture that gesture and ignore vertical movement, resulting in a path-based gesture. If you include touch-screen devices as accessibility supported then these types of interaction need testing with a touch screen as using a mouse in a similar way would not trigger the same browser behavior.

This Success Criterion does not require all functionality to be available through pointing devices, but if it is available to pointer devices then it should not require

path-based gestures. While content authors generally need to provide keyboard commands or other non-pointer mechanisms that perform actions equivalent to complex gestures (see Success Criterion 2.1.1 Keyboard), this is not sufficient to conform to this Success Criterion. That is because some users rely entirely on pointing devices, or find simple pointer inputs much easier to perform and understand than alternatives. For example, a user relying on a head-pointer would find clicking a control to be much more convenient than activating an on-screen keyboard to emulate a keyboard shortcut, and a person who has difficulty memorizing a series of keys (or gestures) may find it much easier to simply click on a labeled control. Therefore, if one or more pointer-based mechanisms are supported, then their benefits should be afforded to users through simple, single-point actions alone.

Single pointer operations include taps and clicks, double-taps and double-clicks, long presses, swiping, dragging, and path-based gestures. Gestures such as "pinch to zoom" or two-finger swipes are *multipoint* gestures, as they require two or more pointer inputs - in this case, two fingers on a touchscreen.

An exception is made for functionality that is inherently and necessarily based on complex paths or multipoint gestures. For example, entering your signature may be inherently path-based (although acknowledging something or confirming your identity need not be).

Gestures that involve dragging in any direction are not in scope for this SC because only the start and end points matter in a dragging operation. However, such gestures do require fine motor control. Authors are encouraged to provide non-dragging methods, for instance, a drag and drop operation could also be achieved by selecting an item (with a tap or keyboard interaction) and then selecting its destination as a second step.

## Benefits

- Users who cannot (accurately) perform path-based pointer gestures - on a touchscreen, or with a mouse - will have alternative means for operating the content.
- Users who cannot perform multi-pointer gestures on a touchscreen (for instance, because they are operating the touchscreen with an alternative input such as a head pointer) will have a single-pointer alternative for operating the content.
- Users who may not understand the custom gesture interaction intended by the

author will be able to rely on simple, frequently used gestures to interact. This can be especially beneficial for users with cognitive or learning disabilities.

### § 6.2.6.3 Pointer Cancellation

For functionality that can be operated using a single pointer, at least one of the following is true:

**No Down-Event**

The down-event of the pointer is not used to execute any part of the function;

**Abort or Undo**

Completion of the function is on the up-event, and a mechanism is available to abort the function before completion or to undo the function after completion;

**Up Reversal**

The up-event reverses any outcome of the preceding down-event;

**Essential**

Completing the function on the down-event is essential.

*Functions that emulate a keyboard or numeric keypad key press are considered essential.*

*This requirement applies to web content that interprets pointer actions (i.e. this does not apply to actions that are required to operate the user agent or assistive technology).*

### § GUIDANCE WHEN APPLYING SUCCESS CRITERION 2.5.2 TO NON-WEB DOCUMENTS AND SOFTWARE

This applies directly as written, and as described in [Intent from Understanding Success Criterion 2.5.2](#) (also provided below), making changes to the notes for non-web documents by replacing “web content” with “content” and for non-web software by replacing “web content” with “non-web software” and “user agent” with “underlying platform software”.

With these substitutions, the notes would read:

(for non-web documents)

#### NOTE 1

Functions that emulate a keyboard or numeric keypad key press are considered essential.

#### NOTE 2

This requirement applies to **[content]** that interprets pointer actions (i.e. this does not apply to actions that are required to operate the user agent or assistive technology).

#### NOTE 3

Content that interprets pointer actions and controls which events are used for executing functionality is less common in documents. An example where a document author could add such functionality is an interactive prototype document created in a software design tool.

(for non-web software)

#### NOTE 4

Functions that emulate a keyboard or numeric keypad key press are considered essential. **[Examples of essential functionality for non-web software are features for meeting environmental energy usage requirements (like waking a device from sleep, power saver mode, and low power state).]**

#### NOTE 5

This requirement applies to **[non-web software]** that interprets pointer actions (i.e. this does not apply to actions that are required to operate the **[underlying platform software]** assistive technology).

## *Intent from Understanding Pointer Cancellation*

The intent of this success criterion is to make it easier for users to prevent accidental or erroneous pointer input. People with various disabilities can inadvertently initiate touch or mouse events with unwanted results. Each of the following subsections roughly aligns with the bullets of this Success Criterion, and outlines a means of allowing users to cancel pointer operations.

## Up-Event activation or completion

The most accessible way to incorporate pointer cancellation is to make activation occur on the up-event.

Up-event activation refers to the activation of a target when the pointer is released. In a touchscreen interaction, when the finger touches a target, the up-event activation only occurs when the finger is lifted while still being within the target boundary. Similarly in mouse interaction, the up-event occurs when the mouse button is released while the cursor is still within the boundary of the initial target set when the mouse button was pressed.

Authors can reduce the problem of users inadvertently triggering an action by using generic platform activation/click events that activate functionality on the up-event. For example, the `click` event in JavaScript triggers on release of the primary mouse button, and is an example of an implicit up-event. Despite its name, the `click` event is device-independent and also works for touch and keyboard interaction.

The preference for up-events is implicit in the Success Criterion wording of the first bullet: “The down-event of the pointer is not used to execute any part of the function.” Authors meet the first bullet by using only the up-event.

## Up-Event Abort or Undo

Where the interaction is equivalent to a simple “click”, up-event activation has a built-in ability to cancel. There is a distinction between when someone touches a screen and when they remove their finger. Similarly, in mouse interaction, there is a difference between pressing and releasing the mouse button. When activation occurs only as the pointer is released, users have the opportunity to Abort (cancel) the activation. Users who have difficulty accurately using a mouse or touchscreen benefit greatly from this basic behaviour. They normally receive visual feedback when an item is pressed. If they discover they have selected the wrong item, they can cancel the action by moving their

pointer or finger away from the target before releasing.

For more complex interactions, such as drag and drop, the down- and up-events may initiate and end a series of actions to complete a process. For example, with drag and drop, the item may be:

1. selected with a press (down-event),
2. moved to a new location, while still being depressed, and
3. released (up-event) to conclude the drop action.

In such a complex action, the need for an Abort or Undo function increases. Designers may elect to confirm the move through something like a confirmation dialog or an undo button, giving the user the ability to Undo the process just completed.

Alternatively, the ability to Abort the action can be achieved if, before completing step 3, the user returns the selected item to its original location and concludes the process there. If other parts of the screen disallow a move, the user can conclude the drag and drop there, effectively nullifying the operation.

## Up Reversal

In other interactions, the down-event may trigger a behaviour which can be reversed when the up-event concludes. Examples of this include press-and-hold actions such as where a transient popup appears (or a video plays) when the user presses on an object (down-event), but the popup (or video) disappears as soon as the user releases the pointer (up-event). Since the up-event reverses the preceding down event, the user is returned to their prior point, and has effectively cancelled the operation.

## Down-Event

Completing the function on the down-event is only permitted when it is essential that the up-event not be used.

The most prevalent essential down-event activation occurs in keyboard emulation. On a physical keyboard, keys by default activate on the down-event -- a letter appears when the key is pressed. If a software keyboard emulator tried to override this expected behaviour by making letters appear when the key is released, the behaviour would be unexpected and would adversely affect expected interaction.

Note that a keyboard has a built-in Backspace or Delete button, which effectively provides an Undo option. Undo is not a requirement of the down-event Essential exception; however, providing an easy way for users to undo any action is a recommended practice (and may be a functional necessity), even where it is not a requirement of this Success Criterion.

Other examples where the timing of an activation is essential and requires the down-event would be:

- An activity that emulates a physical on-press trigger, such as when playing an on-screen piano keyboard. Activation on the up-event would significantly alter the desired behaviour.
- A program for shooting skeets where waiting for the "up" event would invalidate the precise timing necessary for the activation.

#### § 6.2.6.4 Label in Name

For user interface components with labels that include text or images of text, the name contains the text that is presented visually.

*A best practice is to have the text of the label at the start of the name.*

#### § GUIDANCE WHEN APPLYING SUCCESS CRITERION 2.5.3 TO NON-WEB DOCUMENTS AND SOFTWARE

This applies directly as written, and as described in [Intent from Understanding Success Criterion 2.5.3](#) (also provided below).

#### NOTE

See also the discussion on [Closed Functionality](#).

## *Intent from Understanding Label in Name*

The intent of this Success Criterion is to ensure that the words which visually label a component are also the words associated with the component programmatically. This helps ensure that people with disabilities can rely on visible labels as a means to interact with the components.

Most controls are accompanied by a visible text **label**. Those same controls have a programmatic **name**, also known as the [Accessible Name](#). Users typically have a much better experience if the words and characters in the visible label of a control match or are contained within the accessible name. When these match, speech-input users (i.e., users of speech recognition applications) can navigate by speaking the visible text labels of components, such as menus, links, and buttons, that appear on the screen. Sighted users who use text-to-speech (e.g., screen readers) will also have a better experience if the text they hear matches the text they see on the screen.

Note that where a visible text label does not exist for a component, this Success Criterion does not apply to that component.

Where text labels exist and are properly linked to the user interface components through established authoring practices, the label and name will normally match. When they don't match, speech-input users who attempt to use the visible text label as a means of navigation or selection (e.g., "move to Password") will be unsuccessful. The speech-based navigation fails because the visible label spoken by the users does not match (or is not part of) the accessible name that is enabled as a speech-input command. In addition, when the accessible name is different from the visible label, it may function as a hidden command that can be accidentally activated by speech-input users.

Mismatches between visible labels and programmatic names for controls are even more of an issue for speech-input and text-to-speech users who also have cognitive challenges. Mismatches create an extra cognitive load for speech-input users, who must remember to say a speech command that is different from the visible label they see on a control. It also creates extra cognitive load for a text-to-speech user to absorb and understand speech output that does not match the visible label.

In order for the label text and accessible name to be matched, it is first necessary to determine which text on the screen should be considered a label for any given control. There are often multiple text strings in a user interface that may be relevant to a control. However, there are reasons why it is best to conservatively interpret the label as being only the text in close proximity.

Conventionally the label for user interface components is the adjacent text string. The typical positioning for left to right languages is:

- immediately to the left of comboboxes, dropdown lists, text inputs, and other widgets (or in the absence of left-side labels, immediately above and aligned with the left edge of each input)
- immediately to the right of checkboxes and radio buttons
- inside buttons and tabs or immediately below icons serving as buttons

The rationale for some of these conventions is explained in [G162: Positioning labels to maximize predictability of relationships](#).

It is important to bias towards treating only the adjacent text as a label because liberal interpretations of what constitutes a text label can jeopardize the value of this Success Criterion (SC) by lessening predictability. Isolating the label to the single string in close proximity to the component makes it easier for developers, testers, and end users to identify the label targeted for evaluation in this SC. Predictable interpretation of labeling allows users of speech recognition technologies to interact with the element via its conventionally positioned label, and allows users of screen reading technologies to enjoy consistency between the nearby visible label and the announced name of the component.

Note that placeholder text within an input field is not considered an appropriate means of providing a label. The [HTML5 specification](#) states “The placeholder attribute should not be used as an alternative to a `<label>`.” However, it is worth noting that “label” in that HTML5 statement is in code brackets and links to the `label` element. For the purposes of this Label in Name Success Criterion, “label” is not used in such a programmatic sense but is simply referring to a text string in close visual proximity to a component. As such, in the absence of any other nearby text string (as described in the preceding list), if an input contains placeholder text, such text may be a candidate for Label in Name. This is supported both through the accessible name calculation (discussed later) and from the practical sense that where a visible label is not otherwise provided, it is likely that a speech-input user may attempt to use the placeholder text value as a means of interacting with the input.

Text labels “express something in human language”

**Symbolic text characters**

For the purposes of this SC, text should not be considered a visible label if it is used in a symbolic manner, rather than directly “expressing something in human language” as per the definition of text in WCAG. For example, [1.4.5 Images of Text](#) describes considerations for “symbolic text characters.” In the images of text example “B”, “I”, and “ABC” appear on icons in a text editor, where they are meant to symbolize the functions for Bold, Italics, and Spelling, respectively. In such a case, the accessible name should be the function the button serves (e.g., “Spell check” or “Check spelling”), not the visible symbolic characters. A similar text editor is shown in the figure below.



*A detail of the rich text editor in Github, showing a variety of unlabeled icons, including icons resembling text characters.*

Likewise, where an author has used a greater-than symbol (“>”) to mimic the appearance of the right-facing arrow, the text does not convey something in human language. It is a symbol, in this scenario likely meant to mimic the icons used for a “Play” button or a “Next” arrow.

## Punctuation and capitalization

The use of punctuation and capitalization in labels *may* also be considered optional for the same reason. For example, the colon conventionally added at the end of input labels does not express something in human language, and capitals on the first letter of each word in a label do not normally alter the words' meaning. This is particularly relevant in the context of this SC, since it is primarily aimed at users of speech recognition; capitals and most punctuation are frequently ignored when a user speaks a label as a means of interacting with a control.

While it is certainly not an error to include the colon and capitalization in the accessible name, a computed name of “First name” should not be considered a failure of “First Name:”.

First Name:

Likewise, “Next...” visibly shown on a button could have “Next” as the accessible name. When in doubt, where a meaningful visible label exists, match the string exactly for the accessible name.

Next...

## Mathematical expressions and formulae

Mathematical expressions are an exception to the previous subsection about symbolic characters. Math symbols can be used as labels; for example "11×3=33" and "A>B" convey meaning. The label should not be overwritten in the accessible name, and substitutions of words where a formula is used should be avoided since there are multiple ways to express the same equation. For example, making the name "eleven multiplied by three is equivalent to thirty-three" might mean a user who said "eleven times three equals thirty-three" may not match. It is best to leave the formulas as used in the label and count on the user's familiarity with their speech software to achieve a match. Further, converting a mathematical formula label into an accessible name that is a spelled-out equivalent may create issues for translation. The name should match the label's formula text. Note that a consideration for authors is to use the proper symbol in the formula. For instance 11x3 (with a lower or upper case letter X), 11\*3 (with the asterisk symbol), and 11×3 (with the &times; symbol) are all easy for sighted users to interpret as meaning the same formula, but may not all be matched to "11 times 3" by the speech recognition software. The proper operator symbol (in this case the times symbol) should be used.

☐ A>B   ☐ A=B   ☐ A<B

## Accessible Name and Description Computation specification

It is important to understand how the accessible name is derived. The [Accessible Name and Description Computation 1.1](#) and the [HTML Accessibility API Mappings 1.0](#) describe how the accessible name is computed, including which attributes are considered in its calculation, and in what order of preference. If a component has multiple possible attribute values that could be used for its accessible name, only the most preferred of those values will be computed. None of the other, less preferred values will be part of the name. For the most part, existing established programmatic relationships between labels and controls are reinforced by the specification.

Other text displayed on the screen that is correctly coded to meet 1.3.1: Info and Relationships is **not** normally factored into the calculation for the accessible name of a UI component without author intervention (via ARIA labeling techniques). The most common of these are:

- headings and instructions
- group labels for sets of components (i.e., used with `legend/fieldset` or with `role` of `group` or `radiogroup`)

Such textual information may constitute part of the component's *description*. So from both a programmatic viewpoint, and from the conservative tactic of only considering a label to be "adjacent text," neither headings, instructions, nor group 'labels' should normally be considered **labels** for the purpose of this Success Criterion.

It is important to note that the specification allows authors to override the name calculated through native semantics. Both `aria-label` and `aria-labelledby` take precedence in the name calculation, overriding the visible text as the accessible name even when the visible text label is programmatically associated with the control. For this reason, when a visible label already exists, `aria-label` should be avoided or used carefully, and `aria-labelledby` should be used as a supplement with care.

Finally, `aria-describedby` is not included in the Accessible Name computation (instead it is part of the Accessible Description computation). By convention, text associated with a control through `aria-describedby` is announced immediately after the accessible name by screen readers. Therefore, the context of headings, instructions, and group labels can be provided through the accessible description to assist users of screen readers without affecting the experience of those who navigate using speech recognition software.

#### § 6.2.6.5 Motion Actuation

Functionality that can be operated by device motion or user motion can also be operated by user interface components and responding to the motion can be disabled to prevent accidental actuation, except when:

##### **Supported Interface**

The motion is used to operate functionality through an accessibility supported interface;

##### **Essential**

The motion is essential for the function and doing so would invalidate the activity.

#### § GUIDANCE WHEN APPLYING SUCCESS CRITERION 2.5.4 TO NON-WEB DOCUMENTS AND SOFTWARE

This applies directly as written, and as described in [Intent from Understanding Success Criterion 2.5.4](#) (also provided below).

*Intent from Understanding Motion Actuation*

The intent of this success criterion is to ensure that functions triggered by moving a device (for example, shaking or tilting) or by gesturing towards the device (so that sensors like a camera can pick up and interpret the gesturing), can also be operated by more conventional user interface components.

*This criterion concerns input through sensors which respond directly to motions such as gesturing towards, tilting or shaking a device. It does not cover the movement of users through space as registered by geolocation sensors or beacons, or events observed by the device other than intentional gesturing by the user. It also does not cover incidental motion associated with operating a keyboard, pointer, or assistive technology.*

Devices often have sensors that can act as inputs, such as accelerometer and gyroscope sensors on a phone or tablet device. These sensors can allow the user to control something by simply changing the orientation or moving the device in particular ways. In other situations, web content can interpret user gestures via the camera or other sensors to actuate functions. For example, shaking the device might issue an "Undo" command, or a gentle hand wave might be used to move forward or backward in a sequence of pages. Some users with disabilities are not able to operate these device sensors (either not at all, or not precisely enough) because the device is on a fixed mount (perhaps a wheelchair) or due to motor impairments. Therefore, functionality offered through motion must also be available by another mechanism.

In addition, some users may accidentally activate sensors due to tremors or other motor impairments. The user must have the ability to turn off motion actuation to prevent such accidental triggering of functions. Applications may be able to meet this requirement by supporting operating system settings which allow the user to disable motion detection at the system level.

There is an exception where motion is essential for the function or not using motions or gestures would invalidate the activity. Some applications are specifically created to use device sensor data. Examples of content that are exempt from this requirement include a pedometer that relies on device motion to count steps.

## Benefits

- This Success Criterion helps people who may be unable to perform particular motions (such as tilting, shaking, or gesturing) because the device may be

mounted or users may be physically unable to perform the necessary movement. This success criterion ensures that users can still operate all functionality by other means such as touch or via assistive technologies.

- Other users will benefit in situations where they are unable to move their devices.

#### § 6.2.6.6 Dragging Movements

All functionality that uses a dragging movement for operation can be achieved by a single pointer without dragging, unless dragging is essential or the functionality is determined by the user agent and not modified by the author.

*This requirement applies to web content that interprets pointer actions (i.e. this does not apply to actions that are required to operate the user agent or assistive technology).*

#### § GUIDANCE WHEN APPLYING SUCCESS CRITERION 2.5.7 TO NON-WEB DOCUMENTS AND SOFTWARE

##### EDITOR'S NOTE

This section is to be developed by the WCAG2ICT Task Force.

#### *Intent from Understanding Dragging Movements*

The intent of this Success Criterion is to ensure functionality that uses a dragging movement has another single pointer mode of operation without the need for the dexterity required to drag elements.

Some people cannot perform dragging movements in a precise manner. Others use a specialized or adapted input device, such as a trackball, head pointer, eye-gaze system, or speech-controlled mouse emulator, which may make dragging cumbersome and error-prone.

When an interface implements functionality that uses dragging movements, users perform four discrete actions:

1. tap or click to establish a starting point, then
2. press and hold that contact while...
3. performing a repositioning of the pointer, before...
4. releasing the pointer at the end point.

Not all users can accurately press and hold that contact while also repositioning the pointer. An alternative method must be provided so that users with mobility impairments who use a pointer (mouse, pen, or touch contact) can use the functionality.

This requirement is separate from keyboard accessibility because people using a touch screen device may not use a physical keyboard. Keyboard specific interactions such as tabbing or arrow keys may not be possible when encountering a drag and drop control. Note, however, that providing a text input can be an acceptable single-pointer alternative to dragging. For example, an input beside a slider could allow any user to enter a precise value for the slider. In such a situation, the on-screen keyboard that appears for touch users offers a single-pointer means of entering an alphanumeric value.

This criterion does not apply to scrolling enabled by the user-agent. Scrolling a page is not in scope, nor is using a technique such as [CSS overflow](#) to make a section of content scrollable.

## Relationship to other requirements

Success Criteria 2.1.1 Keyboard and 2.1.3 Keyboard (No Exception) require dragging features to be keyboard accessible. However, achieving keyboard equivalence for a

dragging operation does not automatically meet this Success Criterion. It is possible to create an interface that works with dragging and keyboard controls that does not work using only clicks or taps. While many designs can be created for a dragging alternative which address both keyboard accessibility and operability by single pointer operation, the two requirements should be assessed independently.

This Success Criterion applies to dragging movements as opposed to pointer gestures, which are covered in [Success Criterion 2.5.1 Pointer Gestures](#). Pointer gestures include directional path-based as well as multi-point gestures. In contrast, for dragging movements, only the start and end point of the movement matters, not the actual path.

Additional examples are selection rectangles that set the first x/y rectangle coordinate at the pointer position via a pointer down-event, and the second x/y coordinate, after a dragging movement, at the next up-event. A similar example is a connecting line drawn between two different items on the screen, as in an allocation test where users are required to draw a line between questions and corresponding answers. In these cases, the dragging movement requires an alternative way to accomplish the same action that does not rely on the dragging movement. For example, two separate single tap or click actions may define the rectangle coordinates or the start and end points of a connecting line.

## Alternatives for dragging movements on the same page

Where functionality can be executed via dragging movements and an equivalent option exists that allows for single-pointer access without dragging, this Success Criterion is passed. It does not have to be the same component, so long as the functionality is equivalent. An example is a color wheel where a color can be changed by dragging an indicator. In addition, text fields for the numerical input of color values allow the definition of a color without requiring dragging movements. (Note that a text input is considered device agnostic; although the purpose is to enter characters, text entry can take place through voice, pointer or keyboard.)

## Distinguishing dragging movements from path-based pointer gestures

Dragging movements covered in this Success Criterion are pointer interactions where only the start- and endpoints matter. Once the pointer engages with a target, the direction of the dragging movement does not factor into the interaction until the

pointer disengages the target. Since the dragging movement does not have an intermediate point, the dragging movement can go in any direction. Path-based gestures are covered in Success Criterion 2.5.1 Pointer Gestures. For more details, refer to [Understanding Success Criterion 2.5.1 Pointer Gestures](#)

#### § 6.2.6.7 Target Size (Minimum)

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

- **Spacing:** Undersized targets (those less than 24 by 24 CSS pixels) are positioned so that if a 24 CSS pixel diameter circle is centered on the bounding box of each, the circles do not intersect another target or the circle for another undersized target;
- **Equivalent:** The function can be achieved through a different control on the same page that meets this criterion;
- **Inline:** The target is in a sentence or its size is otherwise constrained by the line-height of non-target text;
- **User agent control:** The size of the target is determined by the user agent and is not modified by the author;
- **Essential:** A particular presentation of the target is essential or is legally required for the information being conveyed.

*Targets that allow for values to be selected spatially based on position within the target are considered one target for the purpose of the success criterion. Examples include sliders with granular values, color pickers displaying a gradient of colors, or editable areas where you position the cursor.*

*For inline targets the line-height should be interpreted as perpendicular to the flow of text. For example, in a language displayed vertically, the line-height would be horizontal.*

#### § GUIDANCE WHEN APPLYING SUCCESS CRITERION 2.5.8 TO NON-WEB DOCUMENTS AND SOFTWARE:

This applies directly as written, and as described in [Intent from Understanding Success Criterion 2.5.8](#), replacing "user agent" with "user agent or platform software", and "on the same page" with "in the same non-web document or software".

With these substitutions, it would read:

The size of the [target](#) for [pointer inputs](#) is at least 24 by 24 [CSS pixels](#), except where:

- **Spacing:** Undersized targets (those less than 24 by 24 CSS pixels) are positioned so that if a 24 CSS pixel diameter circle is centered on the [bounding box](#) of each, the circles do not intersect another target or the circle for another undersized target;
- **Equivalent:** The function can be achieved through a different control [\[in the same non-web document or software\]](#) that meets this criterion.
- **Inline:** The target is in a sentence or its size is otherwise constrained by the line-height of non-target text;
- **[\[User agent or platform software\] control:](#)** The size of the target and target offset is determined by the [\[user agent or platform software\]](#) and is not modified by the author;
- **Essential:** A particular [presentation](#) of the target is [essential](#) or is legally required for the information being conveyed;

#### NOTE 1

Targets that allow for values to be selected spatially based on position within the target are considered one target for the purpose of the success criterion. Examples include sliders with granular values, color pickers displaying a gradient of colors, or editable areas where you position the cursor.

#### NOTE 2

For inline targets the line-height should be interpreted as perpendicular to the flow of text. For example, in a language displayed vertically, the line-height would be horizontal.

(for non-web documents)

**NOTE 3**

Some document formats are designed for viewing at a wide range of zoom levels provided by the user agent. However, the commonly available user agents for these formats may lack a consistent base zoom level from which to evaluate this criterion. For such documents, evaluate target sizes at a zoom level that aligns with the intended usage of the content.

(for non-web software)

**NOTE 4**

See also the discussion on [Closed Functionality](#).

*Intent from Understanding Target Size (Minimum)*

The intent of this Success Criterion is to help ensure targets can be easily activated without accidentally activating an adjacent target. Users with dexterity limitations and those who have difficulty with fine motor movement find it difficult to accurately activate small targets when there are other targets that are too close. Providing sufficient size, or sufficient spacing between targets, will reduce the likelihood of accidentally activating the wrong control.

Disabilities addressed by this requirement include hand tremors, spasticity, and quadriplegia. Some people with disabilities use specialized input devices instead of a computer mouse or trackpad. Typically these types of input device do not provide as much accuracy as mainstream pointing devices. Meeting this requirement also ensures that touchscreen interfaces are easier to use.

*This Success Criterion defines a minimum size and, if this can't be met, a minimum spacing. It is still possible to have very small, and difficult to activate, targets and meet the requirements of this Success Criterion, provided that the targets don't have any adjacent targets that are too close. However, using larger target sizes will help many people use targets more easily. As a best practice it is recommended to at least meet the minimum size requirement of the Success Criterion, regardless of spacing. For important links/controls, consider aiming for the stricter [2.5.5 Target Size \(Enhanced\)](#).*

## Exceptions

The requirement is for targets to be at least 24 by 24 CSS pixels in size. There are five exceptions:

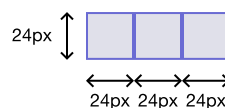
- **Spacing:** Undersized targets (those less than 24 by 24 CSS pixels) are positioned so that if a 24 CSS pixel diameter circle is centered on the bounding box of each, the circles do not intersect another target or the circle for another undersized target.
- **Equivalent:** In cases where a target does not have a size equivalent to 24 by 24 CSS pixels, but there is another control that can achieve the underlying function that *does* meet the minimum size, the target can be excepted based on the "Equivalent" exception.
- **Inline:** The Success Criterion does not apply to inline targets in sentences, or where the size of the target is constrained by the line-height of non-target text. This exception is allowed because text reflow based on viewport size makes it

impossible for authors to anticipate where links may be positioned relative to one another. When multiple links are embedded in blocks of texts in smaller text sizes, guaranteeing that undersized links in adjacent lines of text fulfill the spacing exception (their 24 CSS pixel diameter circle don't intersect any other links or their circles) would require a large line height which can be undesirable in many design contexts.

- **User agent control:** Browsers have default renderings of some controls, such as the days of the month calendar in an `<input type="date">`. As long as the author has not modified the user agent default, the target size for a “User agent control” is excepted.
- **Essential:** If the size and spacing of the targets is fundamental to the information being conveyed, the “Essential” exception applies. For example, in digital maps, the position of pins is analogous to the position of places shown on the map. If there are many pins close together, the spacing between pins and neighboring pins will often be below 24 CSS pixels. It is essential to show the pins at the correct map location, therefore the Essential exception applies. A similar example is an interactive data visualization where targets are necessarily dense. Another example is where jurisdictions legally require online forms to replicate paper forms, which can impose constraints on the size of targets. In such jurisdictions, any legal requirement to replicate small targets can be considered essential. When the essential exception is applicable, authors are strongly encouraged to provide equivalent functionality through alternative means to the extent practical.

## Size requirement

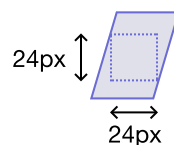
For a target to be "at least 24 by 24 CSS pixels", it must be possible to draw a solid 24 by 24 CSS pixel square, aligned to the horizontal and vertical axis such that the square is completely within the target (does not extend outside the target's area).



*Where targets are a 24 by 24px square (and larger is better), they meet the size requirement of the criterion and pass (image shown to scale - [see the scalable original version](#))*

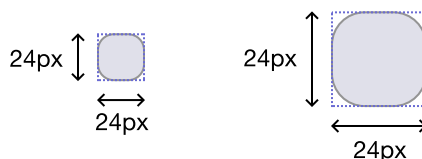
The 24 by 24px square has to be aligned with the page, although the target itself could

be skewed.



*So long as there is a solid 24 by 24px square within the target, it meets the size requirement of the criterion and passes (image shown to scale - [see the scalable original version](#))*

If a target is not large enough to allow for a 24 by 24px square to be drawn inside it, it is considered *undersized*, and does not pass the size requirement of the Success Criterion. However, if it has sufficient space around it without adjacent targets, it may still pass the criterion thanks to the spacing exception (below).



*The rounded corners do not leave sufficient space to draw a 24 by 24px square inside the target, making the target undersized. Depending on the spacing to other targets, it may still pass if it has sufficient clearance (image shown at 1:1 and 2:1 scale - [see the scalable original version](#))*

The requirement is independent of the zoom factor of the page; when users zoom in the CSS pixel size of elements does not change. This means that authors cannot meet it by claiming that the target will have enough spacing or sufficient size if the user zooms into the page.

The requirement does not apply to targets while they are obscured by content displayed as a result of a user interaction or scripted behavior of content, for example:

- interacting with a combobox shows a dropdown list of suggestions
- activating a button displays a modal dialog
- content displays a cookie banner after page load
- content displays a "Take a survey" dialog after some period of user inactivity

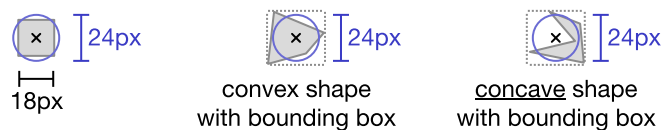
The requirement does however apply to targets in any new content that appears on top of other content.

While the Success Criterion primarily helps touch users by providing target sizing to prevent accidental triggering of adjacent targets, it is also useful for mouse or pen users. It reduces the chances of erroneous activation due to either a tremor or reduced precision, whether because of reduced fine motor control or input imprecision.

## Spacing

When the minimum size for a target is not met, spacing can at least improve the user experience. There is less chance of accidentally activating a neighboring target if a target is not immediately adjacent to another. Touchscreen devices and user agents generally have internal heuristics to identify which link or control is closest to a user's touch interaction - this means that sufficient spacing between targets can work as effectively as a larger target size itself.

When a target is smaller than 24 by 24 CSS pixels, it is *undersized*. In this case, we check if it at least has sufficient *spacing* by drawing a 24 CSS pixel diameter circle over the undersized target, centered on the target's bounding box. For rectangular targets, the bounding box coincides with the target itself – thus, the circle is placed on the center of the target. If the target is *not* rectangular – for instance, if the target is clipped, has rounded corners, or if it's a more complex clickable SVG shape – we need to first determine the bounding box, and then find the box's center. Note that for concave shapes, the center of the bounding box may be outside of the target itself. Now, we center the circle on the center of the bounding box.

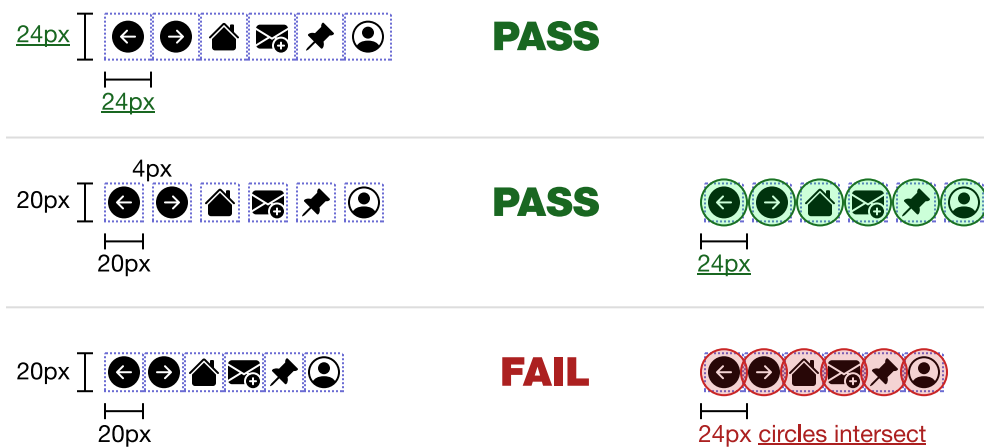


*For a square/rectangular target, the 24 CSS pixel diameter circle is centered on the target itself. For convex and concave targets, it is centered on the bounding box of the shape. Note the concave target, where in this case the center of the bounding box is outside of the actual target (image shown to scale - [see the scalable original version](#))*

We repeat this process for all adjacent undersized targets. To determine if an

undersized target has sufficient spacing (to pass this Success Criterion's spacing exception), we check that the 24 CSS pixel diameter circle of the target does not intersect another target or the circle of any other adjacent undersized targets.

The following example shows a horizontal row of icon-based buttons. In the top row, the dimensions of each target are 24 by 24 CSS pixels, passing this Success Criterion. In the second row, the same targets are only 20 by 20 CSS pixels, but have a 4 CSS pixel space between them – as the target size is below 24 by 24 CSS pixels, these need to be evaluated against the Success Criterion's spacing exception, and they pass. In the last row, the targets are again 20 by 20 CSS pixels, but have no space between them – these fail the spacing exception, because when drawing the 24 CSS pixel diameter circles over the targets, the circles intersect.



*Three rows of targets, illustrating two ways of meeting this Success Criterion and one way of failing it (image shown to scale - [see the scalable original version](#))*

The next two illustrations show sets of buttons which are only 16 CSS pixels tall. In the first set, there are no targets immediately above or below the buttons, so they pass. In the second illustration, there are further buttons, and they have been stacked on top of one another, resulting in a fail.

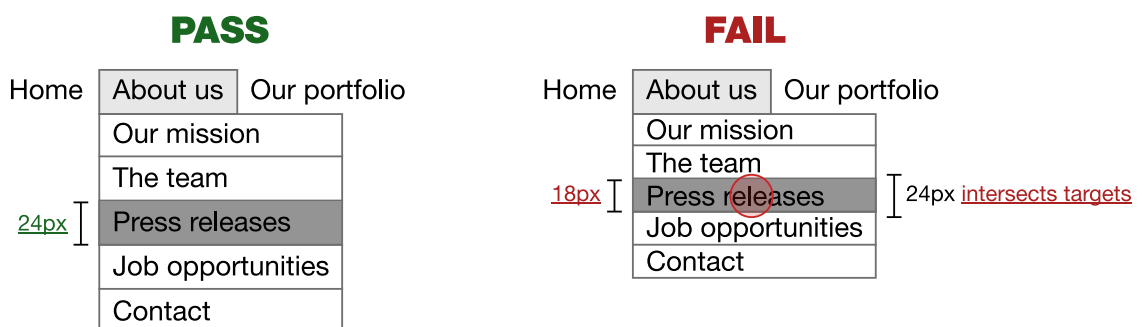


The widths of the buttons with adjacent neighbors is above 24 CSS pixels, while the height is only 16 CSS pixels. However, the lack of adjacent targets above and below means that the targets pass this Success Criterion (image shown to scale - [see the scalable original version](#))



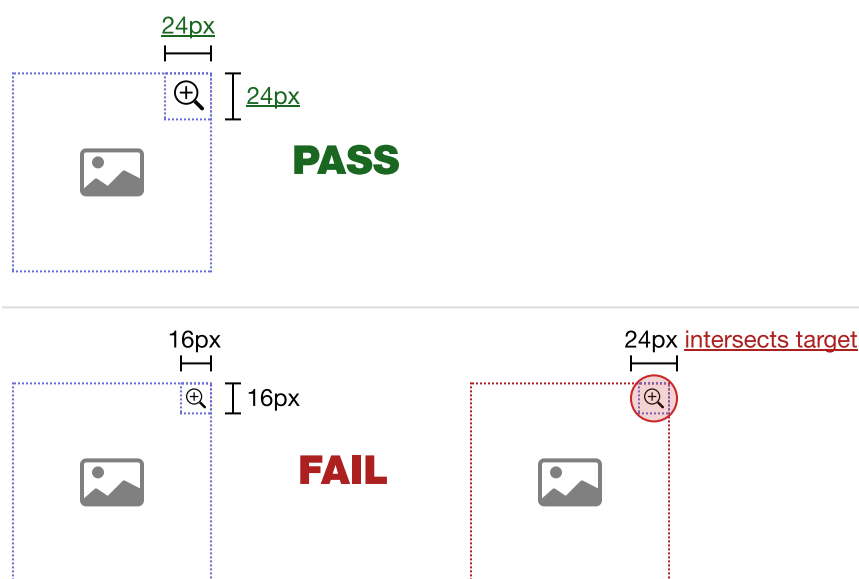
**Fail:** Two rows of buttons, both with a height of only 16 CSS pixels. The rows are close, with only a 1 CSS pixel gap between them, which means that the 24 CSS pixel spacing circles of the targets in one row will intersect the targets (and their circles, depending on their respective widths) in the other line, thus failing the Success Criterion (image shown to scale - [see the scalable original version](#))

The following two illustrations show how menu items can be adjusted to properly meet this requirement. In the first illustration, the “About us” menu has been activated, showing that each of the menu item targets (text and padding) has a 24 CSS pixel height. In the second illustration, the same menu is expanded, but the menu items only achieve 18 CSS pixels in height.



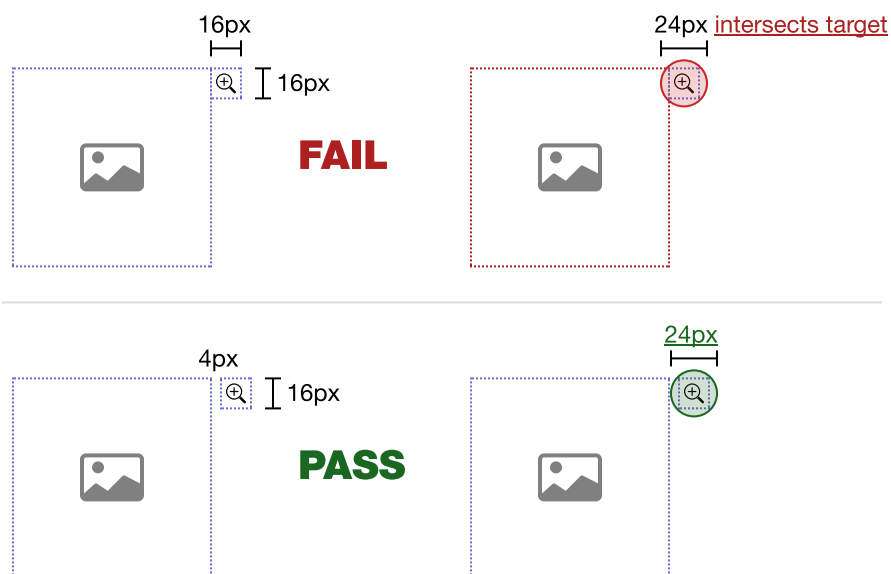
A dropdown menu is shown. The PASS example has menu items which are 24 CSS pixels high. In the FAIL example, the menu items are only 18 CSS pixels high, meaning that the 24 CSS pixel spacing circles of the targets in one row will intersect the adjacent menu item targets and circles (image shown to scale - [see the scalable original version](#))

The following example has one large target (an image that links to a new page related to that image) and a very small second target (a control with a magnifier icon, to open a zoomed-in preview, possibly in a modal, of the image itself). In the top row, the small target overlaps - or, to be more technically accurate, clips - the large target. The small target itself has a size of 24 by 24 CSS pixels, so passes. In the second row, we see that if the second target is any smaller – in this case 16 by 16 CSS pixels – it fails the criterion, as the circle with a 24 CSS pixel diameter we draw over the small target will intersect the large target itself.



Two rows with a small target clipping (overlapping) a large target. In the first row, the 24 by 24 CSS pixel small target passes. In the second row, the 16 by 16 CSS pixel small target fails, since the 24 CSS pixel diameter circle used for undersized targets intersect the large target (image shown to scale - [see the scalable original version](#))

In the following example, we have the same two targets – a large target and a small target. This time, the small target touches/abuts the large target. If the small target is smaller than 24 by 24 CSS pixels, the circle with a 24 CSS pixel diameter we draw over the small target will intersect the large target itself, failing the requirement. The undersized target must be spaced further away from the large target until its circle doesn't intersect the large target.



*In the first row, the 16 by 16 CSS pixel target touching/abutting the large target fails, as its 24 CSS pixel diameter circle used for undersized targets intersects the large target. In the second row we see that the only way the undersized target can pass is by adding a 4 CSS pixel spacing gap between the targets (image shown to scale - [see the scalable original version](#))*

*Users with different disabilities have different needs for control sizes. It can be beneficial to provide an option to increase the active target area without increasing the visible target size. Another option is to provide a mechanism to control the density of layout and thereby change target size or spacing, or both. This can be beneficial for a wide range of users. For example, users with visual field loss may prefer a more condensed layout with smaller sized controls while users with other forms of low vision may prefer large controls.*

## § 6.3 Understandable

Information and the operation of the user interface must be understandable.

## § Guidance When Applying Principle 3 to Non-Web Documents and Software

In WCAG 2.2, the Principles are provided for framing and understanding the success criteria under them but are not required for conformance to WCAG. Principle 3 applies directly as written.

### § **6.3.2 Readable**

Make text content readable and understandable.

### § *Guidance When Applying Guideline 3.1 to Non-Web Documents and Software*

In WCAG 2.2, the Guidelines are provided for framing and understanding the success criteria under them but are not required for conformance to WCAG. Guideline 3.1 applies directly as written.

*Intent from Understanding Readable*

The intent of this guideline is to allow text content to be read by users and by assistive technology, and to ensure that information necessary for understanding it is available.

People with disabilities experience text in many different ways. For some the experience is visual; for some it is auditory; for some it is tactile; for still others it is both visual and auditory. Some users experience great difficulty in recognizing written words yet understand extremely complex and sophisticated documents when the text is read aloud, or when key processes and ideas are illustrated visually or interpreted as sign language. For some users, it is difficult to infer the meaning of a word or phrase from context, especially when the word or phrase is used in an unusual way or has been given a specialized meaning; for these users the ability to read and understand may depend on the availability of specific definitions or the expanded forms of acronyms or abbreviations. User agents, including speech-enabled as well as graphical applications, may be unable to present text correctly unless the language and direction of the text are identified; while these may be minor problems for most users, they can be enormous barriers for users with disabilities. In cases where meaning cannot be determined without pronunciation information (for example, certain Japanese Kanji characters), pronunciation information must be available as well

#### § 6.3.2.2 *Language of Page*

The default human language of each Web page can be programmatically determined.

#### § GUIDANCE WHEN APPLYING SUCCESS CRITERION 3.1.1 TO NON-WEB DOCUMENTS AND SOFTWARE

This applies directly as written, and as described in [Intent from Understanding Success Criterion 3.1.1](#) (also provided below) replacing “each web page” with non-web documents or software.

With these substitutions, it would read:

**3.1.1 Language of Page:** The default [human language](#) of [\[non-web documents or software\]](#) can be [programmatically determined](#). (Level A)

#### NOTE 1

Where software platforms provide a “locale / language” setting, applications that use that setting and render their interface in that “locale / language” would comply with this success criterion. Applications that do not use the platform “locale / language” setting but instead use an [accessibility-supported](#) method for exposing the human language of the [software](#) would also comply with this success criterion. Applications implemented in technologies where [assistive technologies](#) cannot determine the human language and that do not support the platform “locale / language” setting may not be able to meet this success criterion in that locale / language.

#### NOTE 2

See also the discussion on [Closed Functionality](#).

### *Intent from Understanding Language of Page*

The intent of this Success Criterion is to ensure that content developers provide information in the Web page that user agents need to present text and other linguistic content correctly. Both assistive technologies and conventional user agents can render text more accurately when the language of the Web page is identified. Screen readers can load the correct pronunciation rules. Visual browsers can display characters and scripts correctly. Media players can show captions correctly. As a result, users with disabilities will be better able to understand the content.

The default human language of the Web page is the default text-processing language as discussed in [Internationalization Best Practices: Specifying Language in XHTML & HTML Content](#). When a Web page uses several languages, the default text-processing language is the language which is used most. (If several languages are used equally, the first language used should be chosen as the default human language.)

*For multilingual sites targeting Conformance Level A, the Working Group strongly encourages developers to follow Success Criterion 3.1.2 as well even though that is a Level AA Success Criterion.*

### § 6.3.2.3 *Language of Parts*

The human language of each passage or phrase in the content can be programmatically determined except for proper names, technical terms, words of indeterminate language, and words or phrases that have become part of the vernacular of the immediately surrounding text.

## § GUIDANCE WHEN APPLYING SUCCESS CRITERION 3.1.2 TO NON-WEB DOCUMENTS AND SOFTWARE

This applies directly as written, and as described in [Intent from Understanding Success Criterion 3.1.2](#) (also provided below) replacing “content” with “non-web document or software”.

With these substitutions, it would read:

**3.1.2 Language of Parts:** The [human language](#) of each passage or phrase in the **[non-web document or software]** can be [programmatically determined](#) except for proper names, technical terms, words of indeterminate language, and words or phrases that have become part of the vernacular of the immediately surrounding text. (Level AA)

#### NOTE 1

There are some [software](#) and [non-web document](#) technologies where there is no assistive technology supported method for marking the language for the different passages or phrases in the non-web document or software, and it would not be possible to meet this success criterion with those technologies.

#### NOTE 2

See also the discussion on [Closed Functionality](#).

*Intent from Understanding Language of Parts*

The intent of this Success Criterion is to ensure that user agents can correctly present phrases, passages, and in some cases words written in multiple languages. This makes it possible for user agents and assistive technologies to present content according to the presentation and pronunciation rules for that language. This applies to graphical browsers as well as screen readers, braille displays, and other voice browsers.

Both assistive technologies and conventional user agents can render text more accurately if the language of each passage of text is identified. Screen readers can use the pronunciation rules of the language of the text. Visual browsers can display characters and scripts in appropriate ways. This is especially important when switching between languages that read from left to right and languages that read from right to left, or when text is rendered in a language that uses a different alphabet. Users with disabilities who know all the languages used in the Web page will be better able to understand the content when each passage is rendered appropriately.

When no other language has been specified for a phrase or passage of text, its human language is the default human language of the Web page (see [Success Criterion 3.1.1](#)). So the human language of all content in single language documents can be programmatically determined.

Individual words or phrases in one language can become part of another language. For example, "rendezvous" is a French word that has been adopted in English, appears in English dictionaries, and is properly pronounced by English screen readers. Hence a passage of English text may contain the word "rendezvous" without specifying that its human language is French and still satisfy this Success Criterion. Frequently, when the human language of text appears to be changing for a single word, that word has become part of the language of the surrounding text. Because this is so common in some languages, single words should be considered part of the language of the surrounding text unless it is clear that a change in language was intended. If there is doubt whether a change in language is intended, consider whether the word would be pronounced the same (except for accent or intonation) in the language of the immediately surrounding text.

Most professions require frequent use of technical terms which may originate from a foreign language. Such terms are usually not translated to all languages. The universal nature of technical terms also facilitate communication between professionals.

Some common examples of technical terms include: Homo sapiens, Alpha Centauri, hertz, and habeas corpus.

Identifying changes in language is important for a number of reasons:

- It allows braille translation software to follow changes in language, e.g., substitute control codes for accented characters, and insert control codes necessary to prevent erroneous creation of Grade 2 braille contractions.
- Speech synthesizers that support multiple languages will be able to speak the text in the appropriate accent with proper pronunciation. If changes are not marked, the synthesizer will try its best to speak the words in the default language it works in. Thus, the French word for car, "voiture" would be pronounced "voyture" by a speech synthesizer that uses English as its default language.
- Marking changes in language can benefit future developments in technology, for example users who are unable to translate between languages themselves will be able to use machines to translate unfamiliar languages.
- Marking changes in language can also assist user agents in providing definitions using a dictionary.

### § 6.3.3 Predictable

Make Web pages appear and operate in predictable ways.

#### § *Guidance When Applying Guideline 3.2 to Non-Web Documents and Software*

In WCAG 2.2, the Guidelines are provided for framing and understanding the success criteria under them but are not required for conformance to WCAG. Guideline 3.2 applies directly as written, replacing “web pages” with “non-web documents or software”.

With this substitution, this guideline would read:

Guideline 3.2 Predictable: Make **[non-web documents or software]** appear and operate in predictable ways.

*Intent from Understanding Predictable*

The intent of this Guideline is to help users with disabilities by presenting content in a predictable order from Web page to Web page and by making the behavior of functional and interactive components predictable. It is difficult for some users to form an overview of the Web page: screen readers present content as a one-dimensional stream of synthetic speech that makes it difficult to understand spatial relationships. Users with cognitive limitations may become confused if components appear in different places on different pages.

For example, people who use screen magnifiers see only part of the screen at any point in time; a consistent layout makes it easier for them to find navigation bars and other components. Placing repeated components in the same relative order within a set of Web pages allows users with reading disabilities to focus on an area of the screen rather than spending additional time decoding the text of each link. Users with limited use of their hands can more easily determine how to complete their tasks using the fewest keystrokes.

#### § 6.3.3.2 *On Focus*

When any user interface component receives focus, it does not initiate a change of context.

#### § GUIDANCE WHEN APPLYING SUCCESS CRITERION 3.2.1 TO NON-WEB DOCUMENTS AND SOFTWARE

This applies directly as written, and as described in [Intent from Understanding Success Criterion 3.2.1](#) (also provided below).

## NOTE

Some compound documents and their user agents are designed to provide significantly different viewing and editing functionality depending upon what portion of the compound document is being interacted with (e.g. a presentation that contains an embedded spreadsheet, where the menus and toolbars of the user agent change depending upon whether the user is interacting with the presentation content, or the embedded spreadsheet content). If the user uses a mechanism other than putting focus on that portion of the compound document with which they mean to interact (e.g. by a menu choice or special keyboard gesture), any resulting [change of context](#) wouldn't be subject to this success criterion because it was not caused by a change of focus.

### *Intent from Understanding On Focus*

The intent of this Success Criterion is to ensure that functionality is predictable as visitors navigate their way through a document. Any component that is able to trigger an event when it receives focus must not change the context. Examples of changing context when a component receives focus include, but are not limited to:

- forms submitted automatically when a component receives focus;
- new windows launched when a component receives focus;
- focus is changed to another component when that component receives focus;

Focus may be moved to a control either via the keyboard (e.g. tabbing to a control) or the mouse (e.g. clicking on a text field). Moving the mouse over a control does not move the focus unless scripting implements this behavior. Note that for some types of controls, clicking on a control may also activate the control (e.g. button), which may, in turn, initiate a change in context.

*What is meant by "component" here is also sometimes called "user interface element" or "user interface component".*

#### § 6.3.3.3 On Input

Changing the setting of any user interface component does not automatically cause a change of context unless the user has been advised of the behavior before using the component.

## § GUIDANCE WHEN APPLYING SUCCESS CRITERION 3.2.2 TO NON-WEB DOCUMENTS AND SOFTWARE

This applies directly as written, and as described in [Intent from Understanding Success Criterion 3.2.2](#) (also provided below).

### *Intent from Understanding On Input*

The intent of this Success Criterion is to ensure that entering data or selecting a form control has predictable effects. Changing the setting of any user interface component is changing some aspect in the control that will persist when the user is no longer interacting with it. So checking a checkbox, entering text into a text field, or changing the selected option in a list control changes its setting, but activating a link or a button does not. Changes in context can confuse users who do not easily perceive the change or are easily distracted by changes. Changes of context are appropriate only when it is clear that such a change will happen in response to the user's action.

*This Success Criterion covers changes in context due to changing the setting of a control. Clicking on links or tabs in a tab control is activating the control, not changing the setting of that control.*

*What is meant by "component" and "user interface component" here is also sometimes called "user interface element".*

## § 6.3.3.4 Consistent Navigation

Navigational mechanisms that are repeated on multiple Web pages within a set of Web pages occur in the same relative order each time they are repeated, unless a change is initiated by the user.

## § GUIDANCE WHEN APPLYING SUCCESS CRITERION 3.2.3 TO NON-WEB DOCUMENTS AND SOFTWARE

This applies directly as written and described in [Intent from Understanding Success Criterion 3.2.3](#) (also provided below), replacing “set of Web pages” with “set of non-web documents” and “set of software programs”.

With these substitutions, this success criterion would read:

(for non-web documents)

**3.2.3 Consistent Navigation:** Navigational mechanisms that are repeated on multiple **[non-web documents]** within a **[set of non-web documents]** occur in the [same relative order](#) each time they are repeated, unless a change is initiated by the user.

(for software programs)

**3.2.3 Consistent Navigation:** Navigational mechanisms that are repeated on multiple **[software programs]** within a **[set of software programs]** occur in the [same relative order](#) each time they are repeated, unless a change is initiated by the user.

### NOTE 1

See [set of documents](#) and [set of software programs](#) in the Key Terms section of the Introduction to determine when a group of documents or software programs is considered a set for this success criterion. (Sets of software that meet this definition appear to be extremely rare.)

## NOTE 2

Although not required by this success criterion, ensuring that navigation elements have consistent order when repeated *within* non-web documents or software programs directly addresses user needs identified in the Intent section for this Success Criterion, and is generally considered best practice.

### *Intent from Understanding Consistent Navigation*

The intent of this Success Criterion is to encourage the use of consistent presentation and layout for users who interact with repeated content within a set of Web pages and need to locate specific information or functionality more than once. Individuals with low vision who use screen magnification to display a small portion of the screen at a time often use visual cues and page boundaries to quickly locate repeated content. Presenting repeated content in the same order is also important for visual users who use spatial memory or visual cues within the design to locate repeated content.

It is important to note that the use of the phrase "same order" in this section is not meant to imply that subnavigation menus cannot be used or that blocks of secondary navigation or page structure cannot be used. Instead, this Success Criterion is intended to assist users who interact with repeated content across Web pages to be able to predict the location of the content they are looking for and find it more quickly when they encounter it again.

Users may initiate a change in the order by using adaptive user agents or by setting preferences so that the information is presented in a way that is most useful to them.

### § 6.3.3.5 *Consistent Identification*

Components that have the same functionality within a set of Web pages are identified consistently.

## § GUIDANCE WHEN APPLYING SUCCESS CRITERION 3.2.4 TO NON-WEB DOCUMENTS AND SOFTWARE

This applies directly as written and described in [Intent from Understanding Success Criterion 3.2.4](#) (also provided below), replacing “set of web pages” with “set of non-web documents” and “set of software programs”.

With these substitutions, this success criterion would read:

(for non-web documents)

**3.2.4 Consistent Identification:** Components that have the [same functionality](#) within a **[set of non-web documents]** are identified consistently.

(for programs)

**3.2.4 Consistent Identification:** Components that have the [same functionality](#) within a **[set of software programs]** are identified consistently.

### NOTE 1

See [set of documents](#) and [set of software programs](#) in the Key Terms section of the Introduction to determine when a group of documents or software programs is considered a set for this success criterion. (Sets of software that meet this definition appear to be extremely rare.)

### NOTE 2

Although not required by this success criterion, ensuring that component identification be consistent when they occur more than once *within* non-web documents or software programs directly addresses user needs identified in the Intent section for this Success Criterion, and is generally considered best practice.

*Intent from Understanding Consistent Identification*

The intent of this Success Criterion is to ensure consistent identification of functional components that appear repeatedly within a set of Web pages. A strategy that people who use screen readers use when operating a Web site is to rely heavily on their familiarity with functions that may appear on different Web pages. If identical functions have different labels (or, more generally, a different [accessible name](#)) on different Web pages, the site will be considerably more difficult to use. It may also be confusing and increase the cognitive load for people with cognitive limitations. Therefore, consistent labeling will help.

This consistency extends to the text alternatives. If icons or other non-text items have the same functionality, then their text alternatives should be consistent as well.

If there are two components on a web page that both have the same functionality as a component on another page in a set of web pages, then all 3 must be consistent. Hence the two on the same page will be consistent.

While it is desirable and best practice always to be consistent within a single web page, 3.2.4 only addresses consistency within a set of web pages where something is repeated on more than one page in the set.

#### § 6.3.3.6 *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 order relative 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.

*Help mechanisms may be provided directly on the page, or may be provided via a direct link to a different page containing the information.*

*For this Success Criterion, "the same order relative to other page content" can be thought of as how the content is ordered when the page is serialized. The visual position of a help mechanism is likely to be consistent across pages for the same page variation (e.g., CSS break-point). The user can initiate a change, such as changing the page's zoom or orientation, which may trigger a different page variation. This criterion is concerned with relative order across pages displayed in the same page variation (e.g., same zoom level and orientation).*

## § GUIDANCE WHEN APPLYING SUCCESS CRITERION 3.2.6 TO NON-WEB DOCUMENTS AND SOFTWARE

### EDITOR'S NOTE

This section is to be developed by the WCAG2ICT Task Force.

*Intent from Understanding Consistent Help*

The intent of this Success Criterion is to ensure users can find help for completing tasks on a Web site, when it is available. When the placement of the help mechanism is kept consistent across a set of pages, users looking for help will find it easier to identify. This is distinct from interface-level help, such as contextual help, features like spell checkers, and instructional text in a form.

Locating the help mechanism in a consistent location across pages makes it easier for users to find it. For example, when a mechanism or link is located in the header of one Web page, it will be easier to find if it is in the header of other pages. The help mechanism, such as a contact phone number, may be provided directly on the page, or it may also be a direct link to a contact page. Regardless of which approach is used, the mechanism must be located in the same relative order on each page within the set of pages.

When testing this Success Criterion, it is the help item which is relative to the rest of the content. When testing a page, other content that is present across the set of web pages and is before the help item should be before the help item on this page. Items which are after the help item on other pages should be after the help item on this page.

If the help item is visually in a different location, but in the same serial order, that is not helpful from a user's point of view, but it would not fail this criterion.

The location in a smaller viewport may be different than in a larger viewport, but it is best if the mechanism or link is consistent across a set of web pages. A consistent location, both visually and programmatically, is the most usable.

When having problems completing a task on a Web site (or part of a Web site, what we call a set of Web pages), people with some types of disabilities may not be able to work through the issue without further help. Issues could include difficulty: completing a form, or finding a document or page which provides information required to complete a task.

Without help, some users may abandon the task. They may also fail to correctly complete a task, or they may require assistance from people who do not necessarily keep private information secure.

It is not the intent of this Success Criterion to require authors to provide help / access to help. The Criterion only requires that *when* one of the listed forms of help is available across multiple pages that it be in a consistent location. It does not require authors to provide help information on PDFs or other static documents that may be

available for viewing/download from the Web pages. PDFs and other static documents are not considered part of the "set of web pages" from which they are downloaded.

It is also not the intent of this Success Criterion to require a human be available at all times. Ideally, if the human contact is not available during certain hours or certain days then information would be provided so the user can tell when it will be available.

## Help Mechanisms

Typical help mechanisms include:

- Human contact details such as a phone number, email address, hours of operation.
- Human contact mechanism such as a messaging system, chat client, contact form, social media channel.
- Self-help option such as an up-to-date Frequently Asked Questions, How Do I page, Support page.
- A fully automated contact mechanism such as a chatbot.

The order of the types of help listed in the Success Criterion does not imply priority.

## Support for people with cognitive and learning disabilities

This section is not required by the Consistent Help success criterion, but provides advice related to [Making Content Usable for People with Cognitive and Learning Disabilities](#).

The human contact details enable users to connect with the organization or the part of the organization that can assist with the content. For example, an online jobs / recruitment portal may provide a contact method for the team that supports the recruitment portal and not a catch-all for the entire company. Each layer of contact added prolongs the time before the user will receive help.

The human contact mechanism enables a person to express what they are looking for using their own words. For some with cognitive disabilities, this may be the best way for them to find an answer to their problem.

For pages for which no human support is available it helps if a self-help option says that no human support is available. Self-help options can go beyond allowing the user

to search within the site. Contextual help is still recommended (see [Success Criterion 3.3.5](#) for more information), but a self-help option provides a single location that makes it easier for people with cognitive disabilities to understand what help is available without having to hunt for it. While some people may easily be able to identify that no support would be available for a particular type of Web site, this may not be apparent to some users with disabilities.

Chatbots can work for many people, and particularly for people with cognitive disabilities if they:

- recognize misspelled words,
- provide human contact details if the chatbot is unable to provide a satisfactory response after 3 attempts, and
- can be dismissed with a single interaction, and recalled using a link or button.

This criterion does not require that a site provide a help mechanism. However, when help is available:

- People who may have difficulty locating help are more likely to find it and complete their task.
- Users that experience cognitive fatigue or cognitive shut down will be able to reserve their energy for the task, instead of using it to find support.
- Enabling users (especially those with cognitive disabilities) to find solutions while expressing their question using their own words (for example by interacting with a chatbot) increases their chances of success for completing a task.

Self help methods beyond the site, such as using internet search to find the contact information for an organization, can be too difficult. Further, the user's disability may make it more difficult to find the help available (such as a "contact us" link, phone number, or support page) if the information is not consistently present within a few interactions (e.g., displayed in the header, or via a menu). In addition, for some users with disabilities, struggling to complete a task on a site may cause additional cognitive challenges when searching for help within the site.

When a user is quickly able to find help, they are able to complete the task even if they encounter challenges.

## § 6.3.4 Input Assistance

Help users avoid and correct mistakes.

### § Guidance When Applying Guideline 3.3 to Non-Web Documents and Software

In WCAG 2.2, the Guidelines are provided for framing and understanding the success criteria under them but are not required for conformance to WCAG. Guideline 3.3 applies directly as written.

#### *Intent from Understanding Input Assistance*

Everyone makes mistakes. However, people with some disabilities have more difficulty creating error-free input. In addition, it may be harder for them to detect that they have made an error. Typical error indication methods may not be obvious to them because of a limited field of view, limited color perception, or use of assistive technology. This guideline seeks to reduce the number of serious or irreversible errors that are made, increase the likelihood that all errors will be noticed by the user, and help users understand what they should do to correct an error.

### § 6.3.4.2 Error Identification

If an input error is automatically detected, the item that is in error is identified and the error is described to the user in text.

### § GUIDANCE WHEN APPLYING SUCCESS CRITERION 3.3.1 TO NON-WEB DOCUMENTS AND SOFTWARE

This applies directly as written, and as described in [Intent from Understanding Success](#)

[Criterion 3.3.1](#) (also provided below).

 NOTE

See also the discussion on [Closed Functionality](#).

*Intent from Understanding Error Identification*

The intent of this Success Criterion is to ensure that users are aware that an error has occurred and can determine what is wrong. The error message should be as specific as possible. In the case of an unsuccessful form submission, re-displaying the form and indicating the fields in error is insufficient for some users to perceive that an error has occurred. Screen reader users, for example, will not know there was an error until they encounter one of the indicators. They may abandon the form altogether before encountering the error indicator, thinking that the page simply is not functional. Per the definition in WCAG 2.0, an "input error" is information provided by the user that is not accepted. This includes:

- information that is required by the web page but omitted by the user, or
- information that is provided by the user but that falls outside the required data format or allowed values.

For example:

- the user fails to enter the proper abbreviation in to state, province, region, etc. field;
- the user enters a state abbreviation that is not a valid state;
- the user enters a non existent zip or postal code;
- the user enters a birth date 2 years in the future;
- the user enters alphabetic characters or parentheses into their phone number field that only accepts numbers;
- the user enters a bid that is below the previous bid or the minimum bid increment.

*If a user enters a value that is too high or too low, and the coding on the page automatically changes that value to fall within the allowed range, the user's error would still need to be described to them as required by the success criterion. Such an error description telling the person of the changed value would meet both this success criterion (Error Identification) and [Success Criterion 3.3.3 \(Error Suggestion\)](#).*

The identification and description of an error can be combined with programmatic information that user agents or assistive technologies can use to identify an error and provide error information to the user. For example, certain technologies can specify that the user's input must not fall outside a specific range, or that a form field is required. Currently, few technologies support this kind of programmatic information, but the Success Criterion does not require, nor prevent it.

It is perfectly acceptable to indicate the error in other ways such as image, color etc, in addition to the text description.

See also [3.3.3: Error Suggestion](#).

#### § 6.3.4.3 *Labels or Instructions*

Labels or instructions are provided when content requires user input.

#### § GUIDANCE WHEN APPLYING SUCCESS CRITERION 3.3.2 TO NON-WEB DOCUMENTS AND SOFTWARE

This applies directly as written, and as described in [Intent from Understanding Success Criterion 3.3.2](#) (also provided below).

##### *Intent from Understanding Labels or Instructions*

The intent of this Success Criterion is to have content authors present instructions or labels that identify the controls in a form so that users know what input data is expected. In the case of radio buttons, checkboxes, comboboxes, or similar controls that provide users with options, each option must have an appropriate label so that users know what they are actually selecting. Instructions or labels may also specify data formats for data entry fields, especially if they are out of the customary formats or if there are specific rules for correct input. Content authors may also choose to make such instructions available to users only when the individual control has focus especially when instructions are long and verbose.

The intent of this Success Criterion is not to clutter the page with unnecessary information but to provide important cues and instructions that will benefit people with disabilities. Too much information or instruction can be just as harmful as too little. The goal is to make certain that enough information is provided for the user to accomplish the task without undue confusion or navigation.

This Success Criterion does not require that labels or instructions be correctly marked up, identified, or associated with their respective controls - this aspect is covered separately by [1.3.1: Info and Relationships](#). It is possible for content to pass this Success Criterion (providing relevant labels and instructions) while failing Success Criterion 1.3.1 (if the labels or instructions aren't correctly marked up, identified, or associated).

Further, this Success Criterion does not take into consideration whether or not alternative methods of providing an accessible name or description for form controls and inputs has been used - this aspect is covered separately by [4.1.2: Name, Role and Value](#). It is possible for controls and inputs to have an appropriate accessible name or description (e.g. using `aria-label="..."`) and therefore pass Success Criterion 4.1.2, but to still fail this Success Criterion (if the labels or instructions aren't presented to all users, not just those using assistive technologies).

This Success Criterion does not apply to links or other controls (such as an expand/collapse widget, or similar interactive components) that are not associated with data entry.

While this Success Criterion requires that controls and inputs for data entry and submission have labels, whether or not these labels are sufficiently clear or descriptive is covered separately by [2.4.6: Headings and Labels](#).

#### § 6.3.4.4 Error Suggestion

If an input error is automatically detected and suggestions for correction are known, then the suggestions are provided to the user, unless it would jeopardize the security or purpose of the content.

#### § GUIDANCE WHEN APPLYING SUCCESS CRITERION 3.3.3 TO NON-WEB DOCUMENTS AND SOFTWARE

This applies directly as written, and as described in [Intent from Understanding Success Criterion 3.3.3](#) (also provided below).

##### *Intent from Understanding Error Suggestion*

The intent of this Success Criterion is to ensure that users receive appropriate suggestions for correction of an input error if it is possible. The definition of "input error" says that it is "information provided by the user that is not accepted" by the system. Some examples of information that is not accepted include information that is required but omitted by the user and information that is provided by the user but that falls outside the required data format or allowed values.

Success Criterion 3.3.1 provides for notification of errors. However, persons with cognitive limitations may find it difficult to understand how to correct the errors. People with visual disabilities may not be able to figure out exactly how to correct the error. In the case of an unsuccessful form submission, users may abandon the form because they may be unsure of how to correct the error even though they are aware that it has occurred.

The content author may provide the description of the error, or the user agent may provide the description of the error based on technology-specific, programmatically determined information.

#### § 6.3.4.5 Error Prevention (Legal, Financial, Data)

For Web pages that cause legal commitments or financial transactions for the user to occur, that modify or delete user-controllable data in data storage systems, or that submit user test responses, at least one of the following is true:

**Reversible**

Submissions are reversible.

**Checked**

Data entered by the user is checked for input errors and the user is provided an opportunity to correct them.

**Confirmed**

A mechanism is available for reviewing, confirming, and correcting information before finalizing the submission.

## § GUIDANCE WHEN APPLYING SUCCESS CRITERION 3.3.4 TO NON-WEB DOCUMENTS AND SOFTWARE

This applies directly as written, and as described in [Intent from Understanding Success Criterion 3.3.4](#) (also provided below) replacing “web pages” with “non-web documents or software”.

With this substitution, it would read:

**3.3.4 Error Prevention (Legal, Financial, Data):** For **[non-web documents or software]** that cause [legal commitments](#) or financial transactions for the user to occur, that modify or delete [user-controllable](#) data in data storage systems, or that submit user test responses, at least one of the following is true: (Level AA)

1. **Reversible:** Submissions are reversible.
2. **Checked:** Data entered by the user is checked for [input errors](#) and the user is provided an opportunity to correct them.
3. **Confirmed:** A [mechanism](#) is available for reviewing, confirming, and correcting information before finalizing the submission.

*Intent from Understanding Error Prevention (Legal, Financial, Data)*

The intent of this Success Criterion is to help users with disabilities avoid serious consequences as the result of a mistake when performing an action that cannot be reversed. For example, purchasing non-refundable airline tickets or submitting an order to purchase stock in a brokerage account are financial transactions with serious consequences. If a user has made a mistake on the date of air travel, he or she could end up with a ticket for the wrong day that cannot be exchanged. If the user made a mistake on the number of stock shares to be purchased, he or she could end up purchasing more stock than intended. Both of these types of mistakes involve transactions that take place immediately and cannot be altered afterwards, and can be very costly. Likewise, it may be an unrecoverable error if users unintentionally modify or delete data stored in a database that they later need to access, such as their entire travel profile in a travel services web site. When referring to modification or deletion of 'user controllable' data, the intent is to prevent mass loss of data such as deleting a file or record. It is not the intent to require a confirmation for each save command or the simple creation or editing of documents, records or other data.

Users with disabilities may be more likely to make mistakes. People with reading disabilities may transpose numbers and letters, and those with motor disabilities may hit keys by mistake. Providing the ability to reverse actions allows users to correct a mistake that could result in serious consequences. Providing the ability to review and correct information gives the user an opportunity to detect a mistake before taking an action that has serious consequences.

User-controllable data is user-viewable data that the user can change and/or delete through an intentional action. Examples of the user controlling such data would be updating the phone number and address for the user's account, or deleting a record of past invoices from a website. It does not refer such things as internet logs and search engine monitoring data that the user can't view or interact with directly.

#### § 6.3.4.6 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.

## § GUIDANCE WHEN APPLYING SUCCESS CRITERION 3.3.7 TO NON-WEB DOCUMENTS AND SOFTWARE

### EDITOR'S NOTE

This section is to be developed by the WCAG2ICT Task Force.

*Intent from Understanding Redundant Entry*

The intent of this Success Criterion is to ensure that users can successfully complete multi-step processes. It reduces cognitive effort where information is asked for more than once during a process. It also reduces the need to recall information provided in a previous step.

Information that is required to be remembered for input can pose a significant barrier to users with cognitive or memory difficulties. All users experience a natural gradual mental fatigue as they proceed through steps in a process. This fatigue is accelerated by the stress of recalling information from short-term working memory. Users with learning, and cognitive disabilities are highly susceptible to mental fatigue.

Requiring people to recall information previously entered can cause them to give up or re-enter the same information incorrectly. The autocomplete feature of browsers is not considered sufficient because it is the content (the web site) that needs to provide the stored information for a redundant entry, or avoid asking for the same information again.

This Success Criterion does not add a requirement to store information between sessions. A process is defined on the basis of an activity and is not applicable when a user returns after closing a session or navigating away. However, a process can run across different domains, so if a check-out process includes a 3rd party payment provider, that would be in scope.

The term "available to select" is not prescriptive. The term allows authors to develop techniques where auto-population is not possible. It can include allowing the user to:

- select and populate a field, including from a drop-down;
- select text from the page and copy it into an input;
- tick a checkbox to populate inputs with the same values as previously entered (e.g., my billing address is the same as my shipping address).

Data which is "available to select" would need to be on the same page. Ideally, it would be visible by default and closely associated with the input where the data is required. However, it could be elsewhere on a page, including within a show/hide component.

This Success Criterion does not apply if data is provided by the user with a different method, such as uploading a resume in a document format.

This Success Criterion does not impact [Accessible Authentication](#), for which allowing auto-filling of passwords is a sufficient technique. In that case the filling is performed

by the user's browser. Redundant Entry is asking for the website content to make the previous entry available, but not between sessions or for essential purposes such as asking for a password.

This criterion does not include requirements or exceptions specific to privacy or personally identifiable information (PII), but when implementing techniques such as auto-population, authors should ensure data protection when storing information even temporarily during a process. It is possible to eliminate redundant entry in ways that do not introduce additional privacy risks, but it is also possible that a poor implementation (for meeting this criterion) could leak additional PII.

There are exceptions for:

- Essential uses of input re-entry for things like memory games which would be invalidated if the previous answers were supplied.
- Security measures such as preventing a password string from being shown or copied. When creating a password, it should be a unique and complex string and therefore cannot be validated by the author. If the system requires the user to manually create a password that is not displayed, having users re-validate their new string is allowed as an exception.
- When the previously entered information is no longer valid, it can be requested that the user enter that information again.

#### § 6.3.4.7 Accessible Authentication (Minimum)

A cognitive function test (such as remembering a password or solving a puzzle) is not required for any step in an authentication process unless that step provides at least one of the following:

**Alternative**

Another authentication method that does not rely on a cognitive function test.

**Mechanism**

A mechanism is available to assist the user in completing the cognitive function test.

**Object Recognition**

The cognitive function test is to recognize objects.

**Personal Content**

The cognitive function test is to identify non-text content the user provided to the Web site.

*"Object recognition" and "Personal content" may be represented by images, video, or audio.*

*Examples of mechanisms that satisfy this criterion include:*

- 1. support for password entry by password managers to reduce memory need, and*
- 2. copy and paste to reduce the cognitive burden of re-typing.*

## § GUIDANCE WHEN APPLYING SUCCESS CRITERION 3.3.8 TO NON-WEB DOCUMENTS AND SOFTWARE

**EDITOR'S NOTE**

This section is to be developed by the WCAG2ICT Task Force.

*Intent from Understanding Accessible Authentication (Minimum)*

The purpose of this Success Criterion is to ensure there is an accessible, easy-to-use, and secure method to log in. Most Web sites rely on usernames and passwords for logging in. Memorizing or transcribing a username, password, or one-time verification code places a very high or impossible burden upon people with certain cognitive disabilities.

While Web sites can use the recognition of objects or of non-text content provided by the user to meet this Success Criterion, such techniques do not fully support the cognitive accessibility community and should be avoided if possible. Refer to [Accessible Authentication \(Enhanced\)](#) for guidance to be more inclusive and accessible.

## Cognitive Function Tests

Remembering a site-specific password is a cognitive function test. Such tests are known to be problematic for many people with cognitive disabilities. Whether it is remembering random strings of characters, or a pattern gesture to perform on a touch screen, cognitive function tests will exclude some people. When a cognitive function test is used, at least one other authentication method must be available which is not a cognitive function test.

If there is more than one step in the authentication process, such as with multi-factor authentication, all steps need to comply with this Success Criterion to pass. There needs to be a path through authentication that does not rely on cognitive function tests.

Being able to recover or change the email and password is an important part of authentication. If the user is authenticating with alternative information in order to recover their account, there needs to be a method that is not a cognitive function test.

Many organizations are required to use 2-factor authentication that combines independent sources to confirm a user's identity. These sources can consist of combining authentication through:

- knowledge (e.g., password, letters in a passphrase or memorized swipe path);
- possession (e.g., a verification code generated or received on a device, or scanning of a QR code on an external device);
- biometrics (e.g., fingerprint scanning, facial recognition or keystroke dynamics).

Most knowledge-based authentication methods rely on a cognitive function test, so

mechanisms to assist users must be available. When authentication relies on performing an action on a separate device, it should be possible to complete the action without the need to transcribe information. It may not be possible to know what device-based authentication methods are available to a user; offering a choice of methods can allow them to choose the path that most suits them.

## Authentication Approaches

Web sites can employ username (or email) and password inputs as an authentication method if the author enables the user agent (browsers and third-party password managers) to fill in the fields automatically. Generally, if the login form meets [Success Criterion 1.3.5 Input Purpose](#), and the form controls have an appropriate accessible name in accordance with [Success Criterion 4.1.2 Name, Role, Value](#), the user agent should be able to reliably recognize the fields and automatically fill them in. However, if the user agent is actively blocked from filling in the fields (for instance, by a script), then the page would not pass this criterion because it prevents the mechanism from working.

## Copy and paste

Copy and paste can be relied on to avoid transcription. Users can copy their login credentials from a local source (such as a standalone third-party password manager) and paste it into the username and password fields on a login form, or into a web-based command line interfaces asking for a password. Blocking people from pasting into authentication fields, or using a different format between the copied text and the input field (for example, "Enter the 3rd, 4th, and 6th character of your password"), would force the user to transcribe information and therefore fail this criterion, unless another method is available.

## Two-factor authentication systems (verification codes)

Beyond usernames and passwords, some sites may use two-factor authentication, asking the user to enter a verification code (also called a passcode or one-time password). A service that requires *manual* transcription of a verification code is not compliant. As with usernames and passwords, it must be possible for a user to at least paste the code (such as from a standalone third-party password manager, text message

application, or software-based security key), or to allow user agents to fill in the fields automatically.

There are scenarios where a verification code must be received or generated on a secondary device. For example, authenticating in a web browser on a laptop requires a verification code that is sent as an SMS text message to a mobile phone. However, in most cases, it is possible for the code to then be sent directly to the primary device, where it can then be copied and pasted (for example, by copying the code on the secondary device and emailing it to the primary device, or through the use of a shared cross-device clipboard where copying content on the secondary device makes it available to paste on the primary device). Evaluating whether or not the code can be seamlessly transferred from the secondary device to the primary device is *outside of the scope* for this Success Criterion. For the purpose of evaluating Web content that relies on authentication using these types of secondary device systems, it is assumed that provisions are in place that make the code available in the user's clipboard. Evaluating this criterion therefore only requires verification that the web content does allow pasting the clipboard content in the related authentication challenge field.

Note that two-factor systems that do not rely on codes — including hardware authentication devices (such as YubiKey), secondary applications (either on the same primary device, or on a secondary device) that expect the user to confirm that it is indeed them trying to log in, and authentication methods provided by the user's operating system (such as Windows Hello, or Touch ID/Face ID on macOS and iOS) — are *not* a cognitive function test.

## Object Recognition

If a [CAPTCHA](#) is used as part of an authentication process, there must be a method that does not include a cognitive function test, unless it meets the exception. If the test is based on something the website has set such as remembering or transcribing a word, or recognizing a picture the website provided, that would be a cognitive functional test. Recognizing objects, or a picture the user has provided is a cognitive function test; however, it is excepted at the AA level.

An object in this context means the general English definition ("a material thing that can be seen and touched") and can include vehicles and animals. If the test goes beyond recognition (e.g. multiply the number cats by the number of dogs), that does not meet the exception.

Some forms of object recognition may require an understanding of a particular culture. For example, taxis can appear differently in different locales. This is an issue for many people, including people with disabilities, but it is not considered an accessibility-specific issue.

Some CAPTCHAs and cognitive function tests used for authentication may only appear in certain situations, such as when ad blockers are present, or after repeated incorrect password entry. This criterion applies when these tests are used regardless of whether they are used every time or only triggered by specific scenarios.

There are a number of technologies that can be employed to prevent scripted abuse of the authentication process.

- [1.1.1. Rate-limited Access](#)
- [1.1.2. Client Geo-Location](#)
- [1.1.3. Private Client Authentication](#)

None of these systems are 100% effective. However, they may reduce the likelihood of a CAPTCHA being displayed.

## Personal Content

Personal content is sometimes used as a second factor for authentication. For example, as part of account creation the user would upload a picture, and when logging in they would be asked to select that picture from several possible alternatives. Care must be taken to provide adequate security in this case, since non-legitimate users might be able to guess the correct personal content when presented with a choice.

Text-based personal content does not qualify for this exception as it relies on recall (rather than recognition), and transcription (rather than selecting an item). Whilst picture-based personal content will still be a barrier for some people, text based versions tend to be a much larger barrier.

## Hiding characters

Another factor that can contribute to cognitive load is hiding characters when typing. Although this criterion requires that users do not have to type in (transcribe) a password, there are scenarios where that is necessary such as creating a password to

be saved by a password manager. Providing a feature to optionally show a password can improve the chance of success for some people with cognitive disabilities or those who have difficulties with accurately typing.

## § 6.4 Robust

Content must be robust enough that it can be interpreted by a wide variety of user agents, including assistive technologies.

### § Guidance When Applying Principle 4 to Non-Web Documents and Software

In WCAG 2.2, the Principles are provided for framing and understanding the success criteria under them but are not required for conformance to WCAG. Principle 4 applies directly as written replacing “user agents, including assistive technologies” with “assistive technologies and accessibility features of software”.

With this substitution, it would read:

Principle 4: Robust - Content must be robust enough that it can be interpreted reliably by a wide variety of **[assistive technologies and accessibility features of software]**.

### § 6.4.2 Compatible

Maximize compatibility with current and future user agents, including assistive technologies.

### § Guidance When Applying Guideline 4.1 to Non-Web Documents and Software

In WCAG 2.2, the Guidelines are provided for framing and understanding the success

criteria under them but are not required for conformance to WCAG. Guideline 4.1 applies directly as written, replacing “user agents, including assistive technologies” with “assistive technologies and accessibility features of software”.

With this substitution, it would read:

Guideline 4.1 Compatible: Maximize compatibility with current and future **assistive technologies and accessibility features of software**.

### *Intent from Understanding Compatible*

The purpose of this guideline is to support compatibility with current and future user agents, *especially* assistive technologies (AT). This is done both by 1) ensuring that authors do not do things that would break AT (e.g., poorly formed markup) or circumvent AT (e.g., by using unconventional markup or code) and 2) exposing information in the content in standard ways that assistive technologies can recognize and interact with. Since technologies change quickly, and AT developers have much trouble keeping up with rapidly changing technologies, it is important that content follow conventions and be compatible with APIs so that AT can more easily work with new technologies as they evolve.

#### § 6.4.2.2 Parsing (*Obsolete and removed*)

*This criterion was originally adopted to address problems that assistive technology had directly parsing HTML. Assistive technology no longer has any need to directly parse HTML. Consequently, these problems either no longer exist or are addressed by other criteria. This criterion no longer has utility and is removed.*

#### § GUIDANCE WHEN APPLYING SUCCESS CRITERION 4.1.1 TO NON-WEB DOCUMENTS AND SOFTWARE

## EDITOR'S NOTE

This section is to be updated by the WCAG2ICT Task Force since 4.1.1 Parsing is being made obsolete and removed from WCAG 2.2. This work is planned for the next public working draft of the document.

This applies directly as written, and as described in [Intent from Understanding Success Criterion 4.1.1](#) (also provided below), replacing “In content implemented using markup languages” with “For non-web documents or software that use markup languages, in such a way that the markup is separately exposed and available to assistive technologies and accessibility features of software or to a user-selectable user agent”.

With these substitutions, it would read:

**4.1.1 Parsing: [For non-web documents or software that use markup languages, in such a way that the markup is separately exposed and available to assistive technologies and accessibility features of software or to a user-selectable user agent],** elements have complete start and end tags, elements are nested according to their specifications, elements do not contain duplicate attributes, and any IDs are unique, except where the specifications allow these features. (Level A)

## NOTE 1

Start and end tags that are missing a critical character in their formation, such as a closing angle bracket or a mismatched attribute value quotation mark are not complete.

## NOTE 2

Markup is not always available to [assistive technologies](#) or to user selectable [user agents](#) such as browsers. Software sometimes uses markup languages internally for persistence of the software user interface, in ways where the markup is never available to assistive technology (either directly or through a document object model (DOM)), or to a user agent (such as a browser). In such cases, conformance to this provision would have no impact on accessibility as it can have for web content where it is exposed.

Examples of markup that is separately exposed and available to [assistive technologies](#) and to [user agents](#) include: documents encoded in HTML, ODF, and OOXML. In these examples,

the markup can be parsed entirely in two ways: (a) by assistive technologies which may directly open the document, (b) by assistive technologies using DOM APIs of user agents for these document formats.

Examples of markup used internally for persistence of the software user interface that are never exposed to [assistive technology](#) include but are not limited to: XUL, GladeXML, and FXML. In these examples assistive technology only interacts with the user interface of generated software.

#### NOTE 3

See also the discussion on [Closed Functionality](#).

*Intent from Understanding Parsing (Obsolete and removed)*

This criterion has been removed from WCAG 2.2.

The intent of this Success Criterion was to ensure that user-agents, including assistive technologies, can accurately interpret and parse content. Since WCAG 2.0 was published, the specifications (such as HTML) and browsers have improved their handling of parsing errors. It is also the case that assistive technology used to do their own parsing of markup, but now rely on the browser. For that reason this success criterion has been removed. Many issues that would have failed this criterion will fail [Info and Relationships](#) or [Name, Role, Value](#). Other issues are excepted by the "except where the specification allow these features" part of the criterion.

The following content is left for historical purposes to show the original intent.



Success Criterion [4.1.1 Parsing](#) (Level A): In content implemented using markup languages, elements have complete start and end tags, elements are nested according to their specifications, elements do not contain duplicate attributes, and any IDs are unique, except where the specifications allow these features.

*Start and end tags that are missing a critical character in their formation, such as a closing angle bracket or a mismatched attribute value quotation mark are not complete.*

The intent of this Success Criterion is to ensure that user agents, including assistive technologies, can accurately interpret and parse content. If the content cannot be parsed into a data structure, then different user agents may present it differently or be completely unable to parse it. Some user agents use "repair techniques" to render poorly coded content.

Since repair techniques vary among user agents, authors cannot assume that content will be accurately parsed into a data structure or that it will be rendered correctly by specialized user agents, including assistive technologies, unless the content is created according to the rules defined in the formal grammar for that technology. In markup languages, errors in element and attribute syntax and failure to provide properly nested start/end tags lead to errors that prevent user agents from parsing the content reliably. Therefore, the Success Criterion requires that the content can be parsed using only the rules of the formal grammar.

*The concept of "well formed" is close to what is required here. However, exact parsing requirements vary amongst markup languages, and most non XML-based languages do not explicitly define requirements for well formedness. Therefore, it was necessary to be more explicit in the Success Criterion in order to be generally applicable to markup languages. Because the term "well formed" is only defined in XML, and (because end tags are sometimes optional) valid HTML does not require well formed code, the term is not used in this Success Criterion.*

*With the exception of one Success Criterion ( [1.4.4: Resize Text](#), which specifically mentions that the effect specified by the Success Criterion must be achieved without relying on an assistive technology) authors can meet the Success Criteria with content that assumes use of an assistive technology (or access features in use agents) by the user, where such assistive technologies (or access features in user agents) exist and are available to the user.*

#### § 6.4.2.3 Name, Role, Value

For all user interface components (including but not limited to: form elements, links and components generated by scripts), the name and role can be programmatically determined; states, properties, and values that can be set by the user can be programmatically set; and notification of changes to these items is available to user agents, including assistive technologies.

*This success criterion is primarily for Web authors who develop or script their own user interface components. For example, standard HTML controls already meet this success criterion when used according to specification.*

#### § GUIDANCE WHEN APPLYING SUCCESS CRITERION 4.1.2 TO NON-WEB DOCUMENTS AND SOFTWARE

This applies directly as written, and as described in [Intent from Understanding Success Criterion 4.1.2](#) (also provided below), replacing the note with: "This success criterion is primarily for software developers who develop or use custom user interface components.

For example, standard user interface components on most accessibility-supported platforms already meet this success criterion when used according to specification.”

With this substitution, it would read:

**4.1.2 Name, Role, Value:** For all [user interface components](#) (including but not limited to: form elements, links and components generated by scripts), the [name](#) and [role](#) can be [programmatically determined](#); states, properties, and values that can be set by the user can be [programmatically set](#); and notification of changes to these items is available to [user agents](#), including [assistive technologies](#). (Level A)

#### NOTE 1

**[This success criterion is primarily for software developers who develop or use custom user interface components. Standard user interface components on most accessibility-supported platforms already meet this success criterion when used according to specification.]**

#### NOTE 2

For conforming to this success criterion, it is usually best practice for software user interfaces to use the accessibility services provided by platform software. These accessibility services enable interoperability between software user interfaces and both assistive technologies and accessibility features of software in standardized ways. Most platform accessibility services go beyond programmatic exposure of name and role, and programmatic setting of states, properties and values (and notification of same), and specify additional information that could be exposed and / or set (for instance, a list of the available actions for a given user interface component, and a means to programmatically execute one of the listed actions).

#### NOTE 3

For document formats that support interoperability with assistive technology, standard user interface components often meet this success criterion when used according to the general design and accessibility guidance for the document format.

#### NOTE 4

See also the discussion on [Closed Functionality](#).

### *Intent from Understanding Name, Role, Value*

The intent of this Success Criterion is to ensure that Assistive Technologies (AT) can gather information about, activate (or set) and keep up to date on the status of user interface controls in the content.

When standard controls from accessible technologies are used, this process is straightforward. If the user interface elements are used according to specification the conditions of this provision will be met. (See examples of Success Criterion 4.1.2 below)

If custom controls are created, however, or interface elements are programmed (in code or script) to have a different role and/or function than usual, then additional measures need to be taken to ensure that the controls provide important information to assistive technologies and allow themselves to be controlled by assistive technologies.

A particularly important state of a user interface control is whether or not it has focus. The focus state of a control can be programmatically determined, and notifications about change of focus are sent to user agents and assistive technology. Other examples of user interface control state are whether or not a checkbox or radio button has been selected, or whether or not a collapsible tree or list node is expanded or collapsed.

*Success Criterion 4.1.2 requires a programmatically determinable name for all user interface components. Names may be visible or invisible. Occasionally, the name must be visible, in which case it is identified as a label. Refer to the definition of name and label in the glossary for more information.*

#### § 6.4.2.4 Status Messages

In content implemented using markup languages, status messages can be programmatically determined through role or properties such that they can be presented to the user by assistive technologies without receiving focus.

## § GUIDANCE WHEN APPLYING SUCCESS CRITERION 4.1.3 TO NON-WEB DOCUMENTS AND SOFTWARE

This applies directly as written, and as described in [Intent from Understanding Success Criterion 4.1.3](#) (also provided below) replacing "In content implemented using markup languages" with "In content implemented using markup languages, or that supports status message notifications".

With this substitution, it would read: In **[content implemented using markup languages, or that supports status message notifications]**, [status messages](#) can be [programmatically determined](#) through [role](#) or properties such that they can be presented to the user by [assistive technologies](#) without receiving focus.

### NOTE 1

For [non-web documents](#) and [software](#) that are not implemented using markup languages, there is still a user need to have status messages be programmatically exposed so that they can be presented to the user by assistive technologies without receiving focus. This is typically enabled through the use of accessibility services of the user agent or platform software.

### NOTE 2

See also the discussion on [Closed Functionality](#).

*Intent from Understanding Status Messages*

The intent of this Success Criterion is to make users aware of important changes in content that are not given focus, and to do so in a way that doesn't unnecessarily interrupt their work.

The intended beneficiaries are blind and low vision users of assistive technologies with screen reader capabilities. An additional benefit is that assistive technologies for users with cognitive disabilities may achieve an alternative means of indicating (or even delaying or suppressing) status messages, as preferred by the user.

The scope of this Success Criterion is specific to changes in content that involve status messages. A status message is a defined term in WCAG. There are two main criteria that determine whether something meets the definition of a status message:

1. the message “provides information to the user on the success or results of an action, on the waiting state of an application, on the progress of a process, or on the existence of errors;”
2. the message is not delivered via a change in context.

Information can be added to pages which does not meet the definition of a status message. For example, the list of results obtained from a search are not considered a status update and thus are not covered by this Success Criterion. However, brief text messages displayed *about* the completion or status of the search, such as "Searching...", "18 results returned" or "No results returned" would be status updates if they do not take focus. Examples of status messages are given in the section titled [Status Message Examples](#) below.

This Success Criterion specifically addresses scenarios where new content is added to the page without changing the user's context. Changes of context, by their nature, interrupt the user by taking focus. They are already surfaced by assistive technologies, and so have already met the goal to alert the user to new content. As such, messages that involve changes of context do not need to be considered and are not within the scope of this Success Criterion. Examples of scenarios that add new content by changing the context are given in the section titled [Examples of Changes That Are Not Status Messages](#) below.

## Benefits

- When appropriate roles or properties are assigned to status messages, the new content is spoken by screen readers in such a way as to assist blind and low vision

users. Most sighted users can observe text peripherally added to the viewport. Such content provides additional information without affecting the user's current point of regard. The ability of an assistive technology to announce such new important text content allows more users to benefit from an awareness of the information in an equivalent manner.

- Assigning proper roles or properties to status messages provides possible future uses and personalization opportunities, such as the potential to be exploited by assistive technologies created for users with some cognitive disabilities. Where page authors elect to design additions to the screen which do *not* change the user's context (i.e., take focus), the information is arguably of less importance than something presented using a modal dialog, which must be acknowledged by the user. As such, depending on the user's preferences, an assistive technology may choose to delay, suppress, or transform such messages so a user is not unnecessarily interrupted; or conversely the assistive technology may highlight such messages where the user finds it optimal to do so.

## § 7. Comments on Definitions in WCAG 2.2 Glossary

### EDITOR'S NOTE

Terms introduced in new WCAG 2.1 Success Criteria have been added to this document. Any definitions that existed in WCAG 2.0 that have changed since the 2013 WCAG2ICT have also been updated. The following terms introduced by WCAG 2.2 will be added in the next WCAG2ICT draft: cognitive function test, dragging movement, focus indicator, minimum bounding box, and perimeter.

The following is a complete list of definitions from the WCAG 2.2 glossary. Some items apply to all technologies and do not require additional guidance in this document; guidance on the remainder follows.

### § 7.1 Glossary Items that Apply to All Technologies

The following glossary items apply to all technologies and do not require further interpretation for non-web ICT.

- abbreviation
- alternative to time-based media
- ASCII art
- audio
- audio description
- audio-only
- blinking
- bounding box
- CAPTCHA
- captions
- correct reading sequence
- emergency
- essential
- extended audio description
- flash
- functionality
- human language
- idiom
- image of text
- informative
- jargon
- large scale (text)
- legal commitments
- link purpose
- live
- lower secondary education level
- mechanism
- media alternative for text

- navigated sequentially
- non-text content
- normative
- on a full-screen window
- paused
- pointer input
- prerecorded
- presentation
- primary education level
- process
- programatically determined link context
- pure decoration
- real-time event
- relationships
- relied upon (technologies that are)
- same relative order
- sign language
- sign language interpretation
- single pointer
- specific sensory experience
- state
- status message
- synchronized media
- text
- text alternative
- used in an unusual or restricted way
- user-controllable
- video

- video-only
- visually customized

## § 7.2 Glossary Items Used only in AAA Success Criteria

This document does not provide guidance on applying AAA Success Criteria to non-web ICT, including the following definitions.

- blocks of text
- context-sensitive help
- motion animation
- region
- section
- supplemental content
- user inactivity

## § 7.3 Glossary Items with Specific Guidance

Additional guidance is provided for the following glossary entries from WCAG 2.2 when applying them to non-web documents and software.

### § 7.3.1 accessibility supported

supported by users' assistive technologies as well as the accessibility features in browsers and other user agents

To qualify as an accessibility-supported use of a Web content technology (or feature of a technology), both 1 and 2 must be satisfied for a Web content technology (or feature):

1. **The way that the Web content technology is used must be supported by users' assistive technology (AT).** This means that the way that the technology is used has been tested for interoperability with users' assistive technology in the human language(s) of the content,

**AND**

2. **The Web content technology must have accessibility-supported user agents that are available to users.** This means that at least one of the following four statements is true:

1. The technology is supported natively in widely-distributed user agents that are also accessibility supported (such as HTML and CSS);

**OR**

2. The technology is supported in a widely-distributed plug-in that is also accessibility supported;

**OR**

3. The content is available in a closed environment, such as a university or corporate network, where the user agent required by the technology and used by the organization is also accessibility supported;

**OR**

4. The user agent(s) that support the technology are accessibility supported and are available for download or purchase in a way that:
  - does not cost a person with a disability any more than a person without a disability **and**
  - is as easy to find and obtain for a person with a disability as it is for a person without disabilities.

*The Accessibility Guidelines Working Group and the W3C do not specify which or how much support by assistive technologies there must be for a particular use of a Web technology in order for it to be classified as accessibility supported. (See [Level of Assistive Technology Support Needed for "Accessibility Support"](#).)*

*Web technologies can be used in ways that are not accessibility supported as long as they are not relied upon and the page as a whole meets the conformance requirements, including [Conformance Requirement 4](#) and [Conformance Requirement 5](#).*

*When a Web Technology is used in a way that is "accessibility supported," it does not imply that the entire technology or all uses of the technology are supported. Most technologies, including HTML, lack support for at least one feature or use. Pages conform to WCAG only if the uses of the technology that are accessibility supported can be relied upon to meet WCAG requirements.*

*When citing Web content technologies that have multiple versions, the version(s) supported should be specified.*

*One way for authors to locate uses of a technology that are accessibility supported would be to consult compilations of uses that are documented to be accessibility supported. (See [Understanding Accessibility-Supported Web Technology Uses](#).)*

*Authors, companies, technology vendors, or others may document accessibility-supported ways of using Web content technologies. However, all ways of using technologies in the documentation would need to meet the definition of accessibility-supported Web content technologies above.*

## Guidance When Applying “accessibility supported” to Non-Web Documents and Software

This applies directly as written and as described in the WCAG 2.2 glossary, replacing “browsers and other user agents” with “user agents or other software”, replacing “user agents” with “user agents or other software”, replacing “web content technology” with “non-web document or software technology”, adding “or other software extension” after “plug-in”, and replacing all five of the Notes with a single new Note: “Note: The concepts

behind the five Notes and in Understanding Accessibility Supported are applicable to web technologies. The same or similar factors are applicable for non-web technologies.”

With these substitutions and addition, it would read:

### **accessibility supported**

supported by users' [assistive technologies](#) as well as the accessibility features in **[user agents or other software]**

To qualify as an accessibility-supported use of a **[non-web document or software] technology** (or feature of a technology), both 1 and 2 must be satisfied for a **[non-web document or software]** technology (or feature):

1. **The way that the **[non-web document or software] technology** is used must be supported by users' assistive technology (AT).** This means that the way that the technology is used has been tested for interoperability with users' assistive technology in the [human language\(s\)](#) of the [content](#),
- AND**
2. **The **[non-web document or software] technology** must have accessibility-supported user agents **[or other software]** that are available to users.** This means that at least one of the following four statements is true:
  1. The technology is supported natively in widely-distributed user agents **[or other software]** that are also accessibility supported (such as HTML and CSS);
  - OR**
  2. The technology is supported in a widely-distributed plug-in **[or other software extension]** that is also accessibility supported;
  - OR**
  3. The content is available in a closed environment, such as a university or corporate network, where the user agent **[or other software]** required by the technology and used by the organization is also accessibility supported;
  - OR**
  4. The user agent(s) that support the technology are accessibility supported and are available for download or purchase in a way that:
    - does not cost a person with a disability any more than a person without

a disability **and**

- is as easy to find and obtain for a person with a disability as it is for a person without disabilities.

#### NOTE

**[The concepts behind the five Notes and in Understanding Accessibility Supported are applicable to web technologies. The same or similar factors are applicable for non-web technologies.]**

### § 7.3.2 ambiguous to users in general

the purpose cannot be determined from the link and all information of the Web page presented to the user simultaneously with the link (i.e., readers without disabilities would not know what a link would do until they activated it)

*Example: The word guava in the following sentence "One of the notable exports is guava" is a link. The link could lead to a definition of guava, a chart listing the quantity of guava exported or a photograph of people harvesting guava. Until the link is activated, all readers are unsure and the person with a disability is not at any disadvantage.*

### § Guidance When Applying “ambiguous to users in general” to Non-Web Documents and Software

This applies directly as written and as described in the WCAG 2.2 glossary, replacing “Web page” with “non-web document or software”.

With this substitution, it would read:

#### **ambiguous to users in general**

the purpose cannot be determined from the link and all information of the **[non-web**

**document or software]** presented to the user simultaneously with the link (i.e., readers without disabilities would not know what a link would do until they activated it)

Example: The word guava in the following sentence “One of the notable exports is guava” is a link. The link could lead to a definition of guava, a chart listing the quantity of guava exported or a photograph of people harvesting guava. Until the link is activated, all readers are unsure and the person with a disability is not at any disadvantage.

### § 7.3.3 assistive technology

hardware and/or software that acts as a user agent, or along with a mainstream user agent, to provide functionality to meet the requirements of users with disabilities that go beyond those offered by mainstream user agents

*functionality provided by assistive technology includes alternative presentations (e.g., as synthesized speech or magnified content), alternative input methods (e.g., voice), additional navigation or orientation mechanisms, and content transformations (e.g., to make tables more accessible).*

*Assistive technologies often communicate data and messages with mainstream user agents by using and monitoring APIs.*

*The distinction between mainstream user agents and assistive technologies is not absolute. Many mainstream user agents provide some features to assist individuals with disabilities. The basic difference is that mainstream user agents target broad and diverse audiences that usually include people with and without disabilities. Assistive technologies target narrowly defined populations of users with specific disabilities. The assistance provided by an assistive technology is more specific and appropriate to the needs of its target users. The mainstream user agent may provide important functionality to assistive technologies like retrieving Web content from program objects or parsing markup into identifiable bundles.*

*Example: Assistive technologies that are important in the context of this document include the following:*

- screen magnifiers, and other visual reading assistants, which are used by people with visual, perceptual and physical print disabilities to change text font, size, spacing, color, synchronization with speech, etc. in order to improve the visual readability of rendered text and images;
- screen readers, which are used by people who are blind to read textual information through synthesized speech or braille;
- text-to-speech software, which is used by some people with cognitive, language, and learning disabilities to convert text into synthetic speech;
- speech recognition software, which may be used by people who have some physical disabilities;

- alternative keyboards, which are used by people with certain physical disabilities to simulate the keyboard (including alternate keyboards that use head pointers, single switches, sip/puff and other special input devices.);
- alternative pointing devices, which are used by people with certain physical disabilities to simulate mouse pointing and button activations.

## § Guidance When Applying “assistive technology” to Non-Web Documents and Software

This applies directly as written and as described in the WCAG 2.2 glossary, replacing “acts as a user agent” with “acts stand-alone”, replacing “a mainstream user agent” with “mainstream information and communication technologies (ICT)” (later “mainstream ICT]”), and replacing “Web content” with “content”.

With these substitutions, it would read:

### **assistive technology (as used in this document)**

hardware and/or software that acts **[stand-alone]**, or along with **[mainstream information and communication technologies (ICT)]**, to provide functionality to meet the requirements of users with disabilities that go beyond those offered by **[mainstream ICT]**

#### NOTE 1

Functionality provided by assistive technology includes alternative presentations (e.g., as synthesized speech or magnified content), alternative input methods (e.g., voice), additional navigation or orientation mechanisms, and content transformations (e.g., to make tables more accessible).

#### NOTE 2

Assistive technologies often communicate data and messages with **[mainstream ICTs]** by using and monitoring APIs.

### NOTE 3

The distinction between **[mainstream ICTs]** and assistive technologies is not absolute. Many **[mainstream ICTs]** provide some features to assist individuals with disabilities. The basic difference is that **[mainstream ICTs]** target broad and diverse audiences that usually include people with and without disabilities. Assistive technologies target narrowly defined populations of users with specific disabilities. The assistance provided by an assistive technology is more specific and appropriate to the needs of its target users. The **[mainstream ICT]** may provide important functionality to assistive technologies like retrieving **[content]** from program objects or parsing markup into identifiable bundles.

Example: Assistive technologies that are important in the context of this document include the following:

- screen magnifiers, and other visual reading assistants, which are used by people with visual, perceptual and physical print disabilities to change text font, size, spacing, color, synchronization with speech, etc. in order to improve the visual readability of rendered text and images;
- screen readers, which are used by people who are blind to read textual information through synthesized speech or braille;
- text-to-speech software, which is used by some people with cognitive, language, and learning disabilities to convert text into synthetic speech;
- speech recognition software, which may be used by people who have some physical disabilities;
- alternative keyboards, which are used by people with certain physical disabilities to simulate the keyboard (including alternate keyboards that use head pointers, single switches, sip/puff and other special input devices.);
- alternative pointing devices, which are used by people with certain physical disabilities to simulate mouse pointing and button activations.

## § 7.3.4 changes of context

major changes that, if made without user awareness, can disorient users who are not able to view the entire page simultaneously

Changes in context include changes of:

1. user agent;
2. viewport;
3. focus;
4. content that changes the meaning of the Web page

*A change of content is not always a change of context. Changes in content, such as an expanding outline, dynamic menu, or a tab control do not necessarily change the context, unless they also change one of the above (e.g., focus).*

*Example: Opening a new window, moving focus to a different component, going to a new page (including anything that would look to a user as if they had moved to a new page) or significantly re-arranging the content of a page are examples of changes of context.*

## § Guidance When Applying “changes of context” to Non-Web Documents and Software

This applies directly as written and as described in the WCAG 2.2 glossary, replacing “Web page” and “page” with “non-web document or content presented by software”.

With this substitution, it would read:

### **changes of context**

major changes in the content of the **[non-web document or content presented by software]** that, if made without user awareness, can disorient users who are not able to view the entire **non-web document or content presented by software]** simultaneously

Changes in context include changes of:

1. [user agent](#);

2. [viewport](#);
3. focus;
4. content that changes the meaning of the **[non-web document or content presented by software]**.

#### NOTE

A change of content is not always a change of context. Changes in content, such as an expanding outline, dynamic menu, or a tab control do not necessarily change the context, unless they also change one of the above (e.g., focus).

Example: Opening a new window, moving focus to a different component, going to a new page (including anything that would look to a user as if they had moved to a new page) or significantly re-arranging the content of a page are examples of changes of context.

#### NOTE 1

A change in the user agent might include bringing up a new window, or might be a significant change in the menus and/or toolbars that are displayed and available for interacting with some portion of the document.

### § 7.3.5 conformance

satisfying all the requirements of a given standard, guideline or specification

### § *Guidance When Applying “conformance” to Non-Web Documents and Software*

The guidance in this document does not use the term “conformance”.

See the section [Comments on Conformance](#).

### § **7.3.6 conforming alternate version**

version that

1. conforms at the designated level, and
2. provides all of the same information and functionality in the same human language, and
3. is as up to date as the non-conforming content, and
4. for which at least one of the following is true:
  1. the conforming version can be reached from the non-conforming page via an accessibility-supported mechanism, or
  2. the non-conforming version can only be reached from the conforming version, or
  3. the non-conforming version can only be reached from a conforming page that also provides a mechanism to reach the conforming version

*In this definition, "can only be reached" means that there is some mechanism, such as a conditional redirect, that prevents a user from "reaching" (loading) the non-conforming page unless the user had just come from the conforming version.*

*The alternate version does not need to be matched page for page with the original (e.g., the conforming alternate version may consist of multiple pages).*

*If multiple language versions are available, then conforming alternate versions are required for each language offered.*

*Alternate versions may be provided to accommodate different technology environments or user groups. Each version should be as conformant as possible. One version would need to be fully conformant in order to meet [conformance requirement 1](#).*

*The conforming alternative version does not need to reside within the scope of conformance, or even on the same Web site, as long as it is as freely available as the non-conforming version.*

*Alternate versions should not be confused with supplementary content, which support the original page and enhance comprehension.*

*Setting user preferences within the content to produce a conforming version is an acceptable mechanism for reaching another version as long as the method used to set the preferences is accessibility supported.*

See [Understanding Conforming Alternate Versions](#)

## § *Guidance When Applying “conforming alternate version” to Non-Web Documents and Software*

The guidance in this document does not use the term “conforming alternate version”.

See the section [Comments on Conformance](#).

## § **7.3.7 content**

information and sensory experience to be communicated to the user by means of a user agent, including code or markup that defines the content's structure, presentation, and interactions

## § *Guidance When Applying “content (Web Content)” to Non-Web Documents and Software*

See the guidance on [content in the Key Terms section](#).

## § **7.3.8 contrast ratio**

$(L1 + 0.05) / (L2 + 0.05)$ , where

- L1 is the relative luminance of the lighter of the colors, and
- L2 is the relative luminance of the darker of the colors.

*Contrast ratios can range from 1 to 21 (commonly written 1:1 to 21:1).*

*Because authors do not have control over user settings as to how text is rendered (for example font smoothing or anti-aliasing), the contrast ratio for text can be evaluated with anti-aliasing turned off.*

*For the purpose of Success Criteria 1.4.3 and 1.4.6, contrast is measured with respect to the specified background over which the text is rendered in normal usage. If no background color is specified, then white is assumed.*

*Background color is the specified color of content over which the text is to be rendered in normal usage. It is a failure if no background color is specified when the text color is specified, because the user's default background color is unknown and cannot be evaluated for sufficient contrast. For the same reason, it is a failure if no text color is specified when a background color is specified.*

*When there is a border around the letter, the border can add contrast and would be used in calculating the contrast between the letter and its background. A narrow border around the letter would be used as the letter. A wide border around the letter that fills in the inner details of the letters acts as a halo and would be considered background.*

*WCAG conformance should be evaluated for color pairs specified in the content that an author would expect to appear adjacent in typical presentation. Authors need not consider unusual presentations, such as color changes made by the user agent, except where caused by authors' code.*

This applies directly as written and as described in the WCAG 2.2 glossary.

Because relative luminance is defined such that it cannot directly apply to hardware, please note the text in the introduction which reads: “This document does not comment on hardware aspects of products, non-UI aspects of platforms, or the application of WCAG 2.2 for user-interface components as a category, because the basic constructs on which the WCAG 2.2 and / or its conformance are built do not apply to these.”

### § 7.3.9 css pixel

visual angle of about 0.0213 degrees

A CSS pixel is the canonical unit of measure for all lengths and measurements in CSS. This unit is density-independent, and distinct from actual hardware pixels present in a display. User agents and operating systems should ensure that a CSS pixel is set as closely as possible to the [CSS Values and Units Module Level 3 reference pixel](#) `[!css3-values]`, which takes into account the physical dimensions of the display and the assumed viewing distance (factors that cannot be determined by content authors).

### § *Guidance When Applying “CSS pixel” to Non-Web Documents and Software*

#### EDITOR'S NOTE

The WCAG2ICT guidance for this definition is key to understanding and testing several criteria that use CSS pixels to indicate measurements. Are there platforms where the platform-defined density-independent pixel does not approximate a CSS pixel? Are there other well-defined alternatives that could provide a reliable equivalent or alternative measurement?

This applies directly as written and as described in the WCAG 2.2 glossary.

#### NOTE 1

Non-web software and its accompanying platform software do not use CSS pixel measurements. Therefore, use platform-defined density-independent pixel measurements which approximate the CSS reference pixel. Examples of platform-defined density-independent pixel measurements include: points (pt) for iOS and macOS, density-independent pixels (dp) for Android, and effective pixels (epx) for Windows.

#### NOTE 2

Examples where a density-independent pixel may not be defined in the platform:

- Software designed for specific hardware, such as kiosks or office equipment, where the author knows the physical screen size and, potentially, the pixel density.
- Software, such as streaming apps on smart TV platforms or similar software, where the author may lack information about the physical screen size but may know an appropriate viewing distance or viewing angle.

When there is no platform-defined density-independent pixel measurement, the reference pixel size can be approximated in the following manner:

- Determine a viewing distance that matches the use case and display type. For instance, in the case of a touchscreen, the viewing distance is normally less than the length of an arm, typically around 28 inches (71 cm).
- Calculate the size of the reference pixel: Divide the viewing distance by 2688. The number 2688 is obtained by dividing 28 inches (arm's length) by the derived reference pixel size (1/96 inch).

#### NOTE 3

Most software and devices are usable at more than one viewing distance. However, only viewing distances that are plausible for the product can be considered an appropriate approximation for the reference pixel. For example, in software designed for use with a touchscreen, a visual-angle pixel longer than 0.11 inch (0.28 mm) would not be plausible, because this would signify a viewing distance of more than arm's length.

**NOTE 4**

People with low vision often use devices at less than the standard viewing distance. However, basing the device-independent pixel on a typical viewing distance provides a balance of benefits for users with disabilities. If a longer viewing distance were chosen as the basis for the device-independent pixel, the viewport would be measured with a smaller number of larger pixels, causing Success Criterion 1.4.10 Reflow to be less stringent. If a shorter viewing distance were chosen, user interface components would be measured with a larger number of smaller pixels, causing some success criteria, such as 2.5.8 Target Size, to be less stringent.

**§ 7.3.10 down-event**

platform event that occurs when the trigger stimulus of a pointer is depressed

The down-event may have different names on different platforms, such as "touchstart" or "mousedown".

From the [WCAG 2.2 definition for down-event](#):

**§ Guidance When Applying “down-event” to Non-Web Documents and Software**

This applies directly as written and as described in the WCAG 2.2 glossary.

**NOTE**

The down-event may have different names on different platforms. For example **["PointerPressed" or "mousedown"]**.

**§ 7.3.11 general flash and red flash thresholds**

a flash or rapidly changing image sequence is below the threshold (i.e., content **passes**) if any of the following are true:

1. there are no more than three **general flashes** and / or no more than three **red flashes** within any one-second period; or
2. the combined area of flashes occurring concurrently occupies no more than a total of .006 steradians within any 10 degree visual field on the screen (25% of any 10 degree visual field on the screen) at typical viewing distance

where:

- A **general flash** is defined as a pair of opposing changes in relative luminance of 10% or more of the maximum relative luminance (1.0) where the relative luminance of the darker image is below 0.80; and where "a pair of opposing changes" is an increase followed by a decrease, or a decrease followed by an increase, and
- A **red flash** is defined as any pair of opposing transitions involving a saturated red

*Exception:* Flashing that is a fine, balanced, pattern such as white noise or an alternating checkerboard pattern with "squares" smaller than 0.1 degree (of visual field at typical viewing distance) on a side does not violate the thresholds.

*For general software or Web content, using a 341 x 256 pixel rectangle anywhere on the displayed screen area when the content is viewed at 1024 x 768 pixels will provide a good estimate of a 10 degree visual field for standard screen sizes and viewing distances (e.g., 15-17 inch screen at 22-26 inches). This resolution of 75 - 85 ppi is known to be lower, and thus more conservative than the nominal CSS pixel resolution of 96 ppi in CSS specifications. Higher resolutions displays showing the same rendering of the content yield smaller and safer images so it is lower resolutions that are used to define the thresholds.*

*A transition is the change in relative luminance (or relative luminance/color for red flashing) between adjacent peaks and valleys in a plot of relative luminance (or relative luminance/color for red flashing) measurement against time. A flash consists of two opposing transitions.*

*The new working definition in the field for "**pair of opposing transitions involving a saturated red**" (from WCAG 2.2) is a pair of opposing transitions where, one transition is either to or from a state with a value  $R/(R + G + B)$  that is greater than or equal to 0.8, and the difference between states is more than 0.2 (unitless) in the CIE 1976 UCS chromaticity diagram. [[ISO\_9241-391]]*

*Tools are available that will carry out analysis from video screen capture. However, no tool is necessary to evaluate for this condition if flashing is less than or equal to 3 flashes in any one second. Content automatically passes (see #1 and #2 above).*

## § Guidance When Applying “general flash and red flash thresholds” to Non-Web Documents and Software

This applies directly as written and as described in the WCAG 2.2 glossary.

### NOTE

Because this deals with relative luminance and not luminance, it can only be applied to information on a display, not to hardware sources of light.

## § 7.3.12 input error

information provided by the user that is not accepted

*This includes:*

- 1. Information that is required by the Web page but omitted by the user*
- 2. Information that is provided by the user but that falls outside the required data format or values*

## § *Guidance When Applying “input error” to Non-Web Documents and Software*

This applies directly as written and as described in the WCAG 2.2 glossary, replacing “Web page” with “non-web document or software”.

With this substitution, it would read:

### **input error**

information provided by the user that is not accepted

#### **NOTE**

This includes:

1. Information that is required by the **[non-web document or software]** but omitted by the user
2. Information that is provided by the user but that falls outside the required data format or values

## § **7.3.13 keyboard interface**

interface used by software to obtain keystroke input

*A keyboard interface allows users to provide keystroke input to programs even if the native technology does not contain a keyboard.*

*Example: A touchscreen PDA has a keyboard interface built into its operating system as well as a connector for external keyboards. Applications on the PDA can use the interface to obtain keyboard input either from an external keyboard or from other applications that provide simulated keyboard output, such as handwriting interpreters or speech-to-text applications with "keyboard emulation" functionality.*

*Operation of the application (or parts of the application) through a keyboard-operated mouse emulator, such as MouseKeys, does not qualify as operation through a keyboard interface because operation of the program is through its pointing device interface, not through its keyboard interface.*

## § Guidance When Applying “keyboard interface” to Non-Web Documents and Software

This applies directly as written and as described in the WCAG 2.2 glossary.

Please see the note in the [guidance for Success Criterion 2.1.1](#) that uses this definition and which reads: “This does not imply that software always needs to directly support a keyboard or ‘keyboard interface’. Nor does it imply that software always needs to provide a soft keyboard. Underlying platform software may provide device independent input services to applications that enable operation via a keyboard. Software that supports operation via such platform device independent services would be operable by a keyboard and would comply.”

## § 7.3.14 keyboard shortcut

alternative means of triggering an action by the pressing of one or more keys

## § Guidance When Applying “keyboard shortcut” to Non-Web Documents and Software

This applies directly as written and as described in the WCAG 2.2 glossary.

### NOTE

A key command issued by a long press of a key (2 seconds or more) and other accessibility features provided by the platform are not considered a keyboard shortcut. Such commands often occur when there are limited keys, or no modifier keys, present on a device.

## § 7.3.15 label

text or other component with a text alternative that is presented to a user to identify a component within Web content

*A label is presented to all users whereas the name may be hidden and only exposed by assistive technology. In many (but not all) cases the name and the label are the same.*

*The term label is not limited to the label element in HTML.*

## § Guidance When Applying “label” to Non-Web Documents and Software

This applies directly as written and as described in the WCAG 2.2 glossary, replacing “Web Content” with “content” and adding “or by accessibility features of software” after “assistive technology” in Note 1.

With this substitution, it would read:

**label**

[text](#) or other component with a [text alternative](#) that is presented to a user to identify a component within **[content]**

#### NOTE 1

A label is presented to all users whereas the [name](#) may be hidden and only exposed by assistive technology **[or by accessibility features of software]**. In many (but not all) cases the name and the label are the same.

#### NOTE 2

The term label is not limited to the label element in HTML.

### § 7.3.16 name

text by which software can identify a component within Web content to the user

*The name may be hidden and only exposed by assistive technology, whereas a label is presented to all users. In many (but not all) cases, the label and the name are the same.*

*This is unrelated to the name attribute in HTML.*

### § Guidance When Applying “name” to Non-Web Documents and Software

This applies directly as written and as described in the WCAG 2.2 glossary, replacing “Web content” with “content” and adding “or by accessibility features of software” after “assistive technology” in Note 1.

With this substitution, it would read:

#### **name**

text by which software can identify a component within **[content]** to the user

**NOTE 1**

The name may be hidden and only exposed by assistive technology **[or by accessibility features of software]**, whereas a [label](#) is presented to all users. In many (but not all) cases, the label and the name are the same.

**NOTE 2**

This is unrelated to the name attribute in HTML.

**NOTE 1**

“AccessibleName” (or the corresponding term used in different APIs) of the Accessibility API of the platform is an example of such a name.

### **§ 7.3.17 programmatically determined**

determined by software from author-supplied data provided in a way that different user agents, including assistive technologies, can extract and present this information to users in different modalities

*Example 1: Determined in a markup language from elements and attributes that are accessed directly by commonly available assistive technology.*

*Example 2: Determined from technology-specific data structures in a non-markup language and exposed to assistive technology via an accessibility API that is supported by commonly available assistive technology.*

### **§ Guidance When Applying “programmatically determined” to Non-Web Documents and Software**

This applies directly as written and as described in the WCAG 2.2 glossary, replacing “user agents, including assistive technologies” with “assistive technologies and accessibility features of software” and adding and “accessibility features of software” after “assistive technology”.

With this substitution, it would read:

**programmatically determined (programmatically determinable)**

determined by [software](#) from author-supplied data provided in a way that different **[\[assistive technologies and accessibility features of software\]](#)**, can extract and present this information to users in different modalities

Example 1: Determined in a markup language from elements and attributes that are accessed directly by commonly available assistive technology **[\[and accessibility features of software\]](#)**.

Example 2: Determined from technology-specific data structures in a non-markup language and exposed to assistive technology **[\[and accessibility features of software\]](#)** via an accessibility API that is supported by commonly available assistive technology **[\[and accessibility features of software\]](#)**.

**NOTE**

Software typically enables content to be programmatically determined through the use of [accessibility services of platform software](#). Non-web documents typically enable [content](#) to be programmatically determined through the use of accessibility services of the user agent.

§ **7.3.18 programmatically set**

set by software using methods that are supported by user agents, including assistive technologies

### § Guidance When Applying “programmatically set” to Non-Web Documents and Software

This applies directly as written and as described in the WCAG 2.2 glossary, replacing “user agents, including assistive technologies” with “assistive technologies and accessibility features of software”.

With this substitution, it would read:

#### **programmatically set**

set by software using methods that are supported by **assistive technologies and accessibility features of software**

#### NOTE

Software typically enables [content](#) to be programmatically determined through the use of [accessibility services of platform software](#). Non-web documents typically enable content to be programmatically determined through the use of accessibility services of the user agent.

### § 7.3.19 relative luminance

the relative brightness of any point in a colorspace, normalized to 0 for darkest black and 1 for lightest white

*For the sRGB colorspace, the relative luminance of a color is defined as  $L = 0.2126 * R + 0.7152 * G + 0.0722 * B$  where **R**, **G** and **B** are defined as:*

- *if  $R_{sRGB} \leq 0.04045$  then  $R = R_{sRGB}/12.92$  else  $R = ((R_{sRGB}+0.055)/1.055)^{2.4}$*
- *if  $G_{sRGB} \leq 0.04045$  then  $G = G_{sRGB}/12.92$  else  $G = ((G_{sRGB}+0.055)/1.055)^{2.4}$*
- *if  $B_{sRGB} \leq 0.04045$  then  $B = B_{sRGB}/12.92$  else  $B = ((B_{sRGB}+0.055)/1.055)^{2.4}$*

*and  $R_{sRGB}$ ,  $G_{sRGB}$ , and  $B_{sRGB}$  are defined as:*

- *$R_{sRGB} = R_{8bit}/255$*
- *$G_{sRGB} = G_{8bit}/255$*
- *$B_{sRGB} = B_{8bit}/255$*

*The "^" character is the exponentiation operator. (Formula taken from [\[\[SRGB\]\]](#).)*

*Before May 2021 the value of 0.04045 in the definition was different (0.03928). It was taken from an older version of the specification and has been updated. It has no practical effect on the calculations in the context of these guidelines.*

*Almost all systems used today to view Web content assume sRGB encoding. Unless it is known that another color space will be used to process and display the content, authors should evaluate using sRGB colorspace. If using other color spaces, see [Understanding Success Criterion 1.4.3](#).*

*If dithering occurs after delivery, then the source color value is used. For colors that are dithered at the source, the average values of the colors that are dithered should be used (average R, average G, and average B).*

*Tools are available that automatically do the calculations when testing contrast and flash.*

*A [separate page giving the relative luminance definition using MathML](#) to display the formulas is available.*

## § Guidance When Applying “relative luminance” to Non-Web Documents and Software

This applies directly as written and as described in the WCAG 2.2 glossary, replacing “Web content” with “content”.

With this substitution, it would read:

### **relative luminance**

the relative brightness of any point in a colorspace, normalized to 0 for darkest black and 1 for lightest white

#### **NOTE 1**

For the sRGB colorspace, the relative luminance of a color is defined as  $L = 0.2126 \cdot R + 0.7152 \cdot G + 0.0722 \cdot B$  where  $R$ ,  $G$  and  $B$  are defined as:

- if  $R_{sRGB} \leq 0.03928$  then  $R = R_{sRGB}/12.92$  else  $R = ((R_{sRGB} + 0.055)/1.055)^{2.4}$
- if  $G_{sRGB} \leq 0.03928$  then  $G = G_{sRGB}/12.92$  else  $G = ((G_{sRGB} + 0.055)/1.055)^{2.4}$
- if  $B_{sRGB} \leq 0.03928$  then  $B = B_{sRGB}/12.92$  else  $B = ((B_{sRGB} + 0.055)/1.055)^{2.4}$

and  $R_{sRGB}$ ,  $G_{sRGB}$ , and  $B_{sRGB}$  are defined as:

- $R_{sRGB} = R_{8bit}/255$
- $G_{sRGB} = G_{8bit}/255$
- $B_{sRGB} = B_{8bit}/255$

The “^” character is the exponentiation operator. (Formula taken from [\[sRGB\]](#)).

#### **NOTE 2**

Before May 2021 the value of 0.04045 in the definition was different (0.03928). It was taken from an older version of the specification and has been updated. It has no practical effect on the calculations in the context of these guidelines.

**NOTE 3**

Almost all systems used today to view **[content]** assume sRGB encoding. Unless it is known that another color space will be used to process and display the content, authors should evaluate using sRGB colorspace. If using other color spaces, see [Understanding Success Criterion 1.4.3](#).

**NOTE 4**

If dithering occurs after delivery, then the source color value is used. For colors that are dithered at the source, the average values of the colors that are dithered should be used (average R, average G, and average B).

**NOTE 5**

Tools are available that automatically do the calculations when testing contrast and flash.

**NOTE 6**

A [MathML version of the relative luminance definition](#) is available.

**NOTE 1**

Because relative luminance is defined such that it cannot directly apply to hardware, please note the text in the introduction which reads: “This document does not comment on hardware aspects of products, non-UI aspects of platforms, or the application of WCAG 2.2 for user-interface components as a category, because the basic constructs on which the WCAG 2.2 and / or its conformance are built do not apply to these.”

**§ 7.3.20 role**

text or number by which software can identify the function of a component within Web content

*Example: A number that indicates whether an image functions as a hyperlink, command button, or check box.*

## § Guidance When Applying “role” to Non-Web Documents and Software

This applies directly as written and as described in the WCAG 2.2 glossary, replacing “Web content” with “content”.

With this substitution, it would read:

### **role**

text or number by which software can identify the function of a component within **[content]**

*Example: A number that indicates whether an image functions as a hyperlink, command button, or check box.*

### **NOTE**

“AccessibleRole” (or the corresponding term used in different APIs) of the Accessibility API of the platform is an example of such a role.

## § 7.3.21 same functionality

same result when used

*Example: A submit "search" button on one Web page and a "find" button on another Web page may both have a field to enter a term and list topics in the Web site related to the term submitted. In this case, they would have the same functionality but would not be labeled consistently.*

## § Guidance When Applying “same functionality” to Non-Web Documents and Software

This applies directly as written and as described in the WCAG 2.2 glossary, adding a second example (and numbering the first).

With these substitutions, it would read:

### **same functionality**

same result when used

Example 1: A submit “search” button on one web page and a “find” button on another web page may both have a field to enter a term and list topics in the Web site related to the term submitted. In this case, they would have the same functionality but would not be labeled consistently.

Example 2: A ribbon icon that saves the document that looks like an arrow pointing into a folder in one case, and an arrow pointing into a hard drive in another. In this case as well, they would have the same functionality but would not be labeled consistently.

## § 7.3.22 satisfies a success criterion

the success criterion does not evaluate to 'false' when applied to the page

## § Guidance When Applying “satisfies a success criterion” to Non-Web Documents and Software

The guidance in this document does not use the term “satisfies a success criterion”.

See [Section 6 Comments on Conformance](#).

### § 7.3.23 set of web pages

collection of web pages that share a common purpose and that are created by the same author, group or organization

*Example: Examples include:*

- *a publication which is split across multiple Web pages, where each page contains one chapter or other significant section of the work. The publication is logically a single contiguous unit, and contains navigation features that enable access to the full set of pages.*
- *an e-commerce website shows products in a set of Web pages that all share the same navigation and identification. However, when progressing to the checkout process, the template changes; the navigation and other elements are removed, so the pages in that process are functionally and visually different. The checkout pages are not part of the set of product pages.*
- *a blog on a sub-domain (e.g. [blog.example.com](#)) which has a different navigation and is authored by a distinct set of people from the pages on the primary domain ([example.com](#)).*

*Different language versions would be considered different sets of Web pages.*

## § *Guidance When Applying “set of Web pages” to Non-Web Documents and Software*

See the guidance on [set of documents](#) and [set of software programs](#) in the [Key Terms](#) section.

### NOTE

For success criteria that use the term “set of web pages”, either explicitly or implicitly ([2.4.1](#), [2.4.5](#), [3.2.3](#), and [3.2.4](#)), simply substitute "set of non-web documents" and "set of software programs" when applying this to non-web technologies.

## § **7.3.24 structure**

1. The way the parts of a Web page are organized in relation to each other; and
2. The way a collection of Web pages is organized

## § *Guidance When Applying “structure” to Non-Web Documents and Software*

This applies directly as written and as described in the WCAG 2.2 glossary, replacing “a Web page” with “non-web documents or software” and replacing “collection of Web pages” with “set of documents or set of software programs”.

With these substitutions, it would read:

### **structure**

1. The way the parts of **[non-web documents or software]** are organized in relation to each other; and
2. The way a **[set of documents or set of software programs]** is organized

#### NOTE 1

See the guidance on user [sets of documents](#) and [sets of software programs](#) in the Key Terms section.

#### NOTE 2

“AccessibleRole” (or the corresponding term used in different APIs) of the Accessibility API of the platform is an example of such a role.

### § 7.3.25 style property

property whose value determines the presentation (e.g. font, color, size, location, padding, volume, synthesized speech prosody) of content elements as they are rendered (e.g. onscreen, via loudspeaker, via braille display) by user agents

Style properties can have several origins:

- User agent default styles: The default style property values applied in the absence of any author or user styles. Some web content technologies specify a default rendering, others do not;
- Author styles: Style property values that are set by the author as part of the content (e.g. in-line styles, author style sheets);
- User styles: Style property values that are set by the user (e.g. via user agent interface settings, user style sheets)

### § *Guidance When Applying “style property” to Non-Web Documents and Software*

This applies directly as written and as described in the WCAG 2.2 glossary, replacing “user agent(s)” with “user agent(s) or platform software”, “web content” with “content”, replacing “in-line styles, author style sheets” with “programmatically-set styles”, and replacing “user agent interface settings, user style sheets” with “user agent, platform software or other

software settings".

With these substitutions, it would read:

### **style property**

property whose value determines the presentation (e.g. font, color, size, location, padding, volume, synthesized speech prosody) of content elements as they are rendered (e.g. onscreen, via loudspeaker, via braille display) by **[user agents or platform software]**

Style properties can have several origins:

- **[User agent or platform software] default styles:** The default style property values applied in the absence of any author or user styles. Some **[content]** technologies specify a default rendering, others do not;
- **Author styles:** Style property values that are set by the author as part of the content (e.g. **[programmatically-set styles]**);
- **User styles:** Style property values that are set by the user (e.g. via **[user agent, platform software or other software]** interface settings)

### **§ 7.3.26 target**

region of the display that will accept a pointer action, such as the interactive area of a user interface component

*If two or more targets are overlapping, the overlapping area should not be included in the measurement of the target size, except when the overlapping targets perform the same action or open the same page.*

### **§ Guidance When Applying “target” to Non-Web Documents and Software**

This applies directly as written and as described in the WCAG 2.2 glossary, replacing “page” with “non-web document or content presented by software”.

With this substitution, it would read:

**target**

region of the display that will accept a pointer action, such as the interactive area of a user interface component

**NOTE**

If two or more targets are overlapping, the overlapping area should not be included in the measurement of the target size, except when the overlapping targets perform the same action or open the same **[non-web document or content presented by software]**.

**§ 7.3.27 technology**

mechanism for encoding instructions to be rendered, played or executed by user agents

*As used in these guidelines "Web Technology" and the word "technology" (when used alone) both refer to Web Content Technologies.*

*Web content technologies may include markup languages, data formats, or programming languages that authors may use alone or in combination to create end-user experiences that range from static Web pages to synchronized media presentations to dynamic Web applications.*

*Example: Some common examples of Web content technologies include HTML, CSS, SVG, PNG, PDF, Flash, and JavaScript.*

**§ Guidance When Applying “technology” to Non-Web Documents and Software**

This applies directly as written and as described in the WCAG 2.2 glossary, replacing “web content” with “non-web document or software”, “user agents” with “user agents or other

software”, removing the notes, and replacing the example with “Example: Some common examples of non-web document and software technologies include ODF, OOXML, Java, and C++.”

With these substitutions, it would read:

**technology (\*\*[non-web document or software]\*\*)**

mechanism for encoding instructions to be rendered, played or executed by [user agents or other software].

Example: Some common examples of [non-web document and software technologies include ODF, OOXML, Java, and C++].

### § 7.3.28 up-event

platform event that occurs when the trigger stimulus of a pointer is released

The up-event may have different names on different platforms, such as "touchend" or "mouseup".

### § Guidance When Applying “up-event” to Non-Web Documents and Software

This applies directly as written and as described in the WCAG 2.2 glossary.

#### NOTE

The up-event may have different names on different platforms, such as [“PointerReleased” or “mouseup”].

### § 7.3.29 user agent

any software that retrieves and presents Web content for users

*Example: Web browsers, media players, plug-ins, and other programs — including assistive technologies — that help in retrieving, rendering, and interacting with Web content.*

## § Guidance When Applying “user agent” to Non-Web Documents and Software

See the [guidance on user agent in the Key Terms section](#).

### § 7.3.30 user interface component

a part of the content that is perceived by users as a single control for a distinct function

*Multiple user interface components may be implemented as a single programmatic element. "Components" here is not tied to programming techniques, but rather to what the user perceives as separate controls.*

*User interface components include form elements and links as well as components generated by scripts.*

*What is meant by "component" or "user interface component" here is also sometimes called "user interface element".*

*Example: An applet has a "control" that can be used to move through content by line or page or random access. Since each of these would need to have a name and be settable independently, they would each be a "user interface component."*

## § Guidance When Applying “user interface component” to Non-Web Documents and Software

This applies directly as written and as described in the WCAG 2.2 glossary, replacing the example with “Example: A software program has 2 controls: a text field for entering a file name and a drop down list box for choosing a folder. Each is a user interface component with a name that is settable by the software.”

With this substitution, it would read:

### **user interface component**

a part of the [content](#) that is perceived by users as a single control for a distinct function

#### NOTE 1

Multiple user interface components may be implemented as a single programmatic element. "Components" here is not tied to programming techniques, but rather to what the user perceives as separate controls.

#### NOTE 2

User interface components include form elements and links as well as components generated by scripts.

#### NOTE 3

What is meant by "component" or "user interface component" here is also sometimes called "user interface element".

Example: A [software](#) program has 2 controls: a text field for entering a file name and a drop down list box for choosing a folder. Each is a user interface component with a name that is settable by the software.

### § 7.3.31 viewport

object in which the user agent presents content

*The user agent presents content through one or more viewports. Viewports include windows, frames, loudspeakers, and virtual magnifying glasses. A viewport may contain another viewport (e.g., nested frames). Interface components created by the user agent such as prompts, menus, and alerts are not viewports.*

*This definition is based on [User Agent Accessibility Guidelines 1.0 Glossary](#) [[UAAG10]].*

### § Guidance When Applying “viewport” to Non-Web Documents and Software

This applies directly as written and as described in the WCAG 2.2 glossary, replacing “user agent” with “software”.

With this substitution, it would read:

#### **viewport**

object in which the **[software]** presents [content](#)

#### NOTE 1

The **[software]** presents content through one or more viewports. Viewports include windows, frames, loudspeakers, and virtual magnifying glasses. A viewport may contain another viewport (e.g., nested frames). Interface components created by the **[software]** such as prompts, menus, and alerts are not viewports.

#### NOTE 2

This definition is based on [User Agent Accessibility Guidelines 1.0 Glossary](#).

### § 7.3.32 web page

a non-embedded resource obtained from a single URI using HTTP plus any other resources that are used in the rendering or intended to be rendered together with it by a user agent

*Although any "other resources" would be rendered together with the primary resource, they would not necessarily be rendered simultaneously with each other.*

*For the purposes of conformance with these guidelines, a resource must be "non-embedded" within the scope of conformance to be considered a Web page.*

*Example 1: A Web resource including all embedded images and media.*

*Example 2: A Web mail program built using Asynchronous JavaScript and XML (AJAX). The program lives entirely at <http://example.com/mail>, but includes an inbox, a contacts area and a calendar. Links or buttons are provided that cause the inbox, contacts, or calendar to display, but do not change the URI of the page as a whole.*

*Example 3: A customizable portal site, where users can choose content to display from a set of different content modules.*

*Example 4: When you enter "<http://shopping.example.com/>" in your browser, you enter a movie-like interactive shopping environment where you visually move around in a store dragging products off of the shelves around you and into a visual shopping cart in front of you. Clicking on a product causes it to be demonstrated with a specification sheet floating alongside. This might be a single-page Web site or just one page within a Web site.*

## § Guidance When Applying “Web Page” to Non-Web Documents and Software

This applies directly as written and as described in the WCAG 2.2 glossary.

### NOTE

For those success criteria that use the term “web page”, WCAG2ICT provides specific replacement term(s) for “Web page”.

## § 8. Privacy Considerations

*This section is non-normative.*

This Working Group Note does not introduce any new privacy considerations. Horizontal Review Groups are encourage to provide further feedback during the Horizontal Review process.

## § 9. Security Considerations

*This section is non-normative.*

This Working Group Note does not introduce any new security considerations. Horizontal Review Groups are encourage to provide further feedback during the Horizontal Review process.

## § A. Success Criteria Problematic for Closed Functionality

## EDITOR'S NOTE

This section has been updated to add notes for new WCAG 2.1 criteria that are problematic for Closed Functionality.

In the next public draft, the WCAG2ICT Task Force will remove obsolete criteria and add new WCAG 2.2 criteria if any are problematic for Closed Functionality. The Task Force may also make updates to existing WCAG 2.0 content in this section to reflect new insights learned since the 2013 WCAG2ICT Note.

There are Success Criteria that can be problematic for developers of closed functionality. Some criteria discuss making information available in text (which can be read by assistive technologies), making it “programmatically determinable” (rendered by a user agent and readable by assistive technologies), or doing something else to make content compatible with assistive technologies. Other Success Criteria would apply to systems with closed functionality either if they are partially closed or if they allow for the connection of some types of devices. As an example, Success Criterion 2.1.1 Keyboard would apply to systems which are closed to screen readers, but have a physical keyboard or a connector for standard keyboards. While these criteria, as written, are not suitable for closed functionality, most of them can inform and aid development of built-in features needed to make closed functionality products accessible. For non-web software on closed functionality products, alternate accessibility provisions might be needed to cover the user needs addressed by the following Success Criteria:

- [1.1.1 Non-text Content](#)—requires text or a text alternative;
- [1.2.1 Pre-recorded video](#)—requires a text alternative for time based media;
- [1.2.3 Audio description or Media Alternative](#)—one of the options available to authors for success criterion 1.2.3 is that of providing a media alternative that is text—which necessarily relies on a connected assistive technology to be presented;
- [1.3.1 Info and Relationships](#)—requires information in a programmatically determinable form;
- [1.3.2 Meaningful Sequence](#)—requires information in a programmatically determinable form;
- [1.3.5 Identify Input Purpose](#)—requires information in a programmatically determinable form;
- [1.4.4 Resize Text](#)—because the text rendering support in a closed environment may be more limited than the support found in user agents for the Web, meeting Success

Criterion 1.4.4 in a closed environment may place a much heavier burden on the content author;

- [1.4.5 Images of Text](#)—because there is no need to impose a requirement on all closed functionality that text displayed on the screen actually be represented internally as text (as defined by WCAG 2.2), given that there is no interoperability with assistive technology;
- [1.4.10 Reflow](#)—Many closed functionality products do not allow users to modify the viewport or change font sizes, thus there would be no need to impose a requirement on all closed functionality that content is able to reflow. Additionally, many closed functionality products do not display large chunks of text and only have UI controls. In such cases, two-directional scrolling to access the text and UI controls may be considered essential.
- [1.4.11 Non-text Contrast](#)—There are cases where applying this Success Criterion to non-web software on closed functionality products is problematic:
  - When the appearance of the content is determined by the hardware and not modifiable by the software author, the non-web software would meet the exception for this Success Criterion.

#### NOTE 1

Hardware requirements for contrast are out of scope for WCAG2ICT (and this Success Criterion), but do exist in other standards' requirements for closed functionality products (e.g. EN 301 549 and Revised 508 Standards).

- When the color contrast ratio cannot be programmatically measured due to system limitations (e.g. lockdown), precise quantifiable testing of color contrast cannot be performed by a third party. In such cases, the software author would need to confirm that the color combinations used meet the contrast requirement.

#### NOTE 2

Photographs are not sufficient for testing that content meets this Success Criteria. This is because the quality of the lighting, camera, and physical aspects of the hardware display can dramatically affect the ability to capture the content for testing purposes.

- [2.1.1 Keyboard](#)—requires operation via a keyboard interface which allows alternative input devices;
- [2.4.2 Page Titled](#)—where software is an integral part of hardware that provides a single

function, such as a calculator or IP telephone, there is no need for a title;

- [2.5.3 Label in Name](#)—requires information in a programmatically determinable form; specifically, the programmatic name contains the text of the visual label;
- [3.1.1 Language of Page](#)—requires information in a programmatically determinable form;
- [3.1.2 Language of Parts](#)—requires information in a programmatically determinable form;
- [3.3.1 Error Identification](#)—while it's important for errors that can be detected to be described to the user, for closed functionality, the error description doesn't have to be provided in text, as defined in WCAG 2.2;
- [4.1.1 Parsing](#)—the [Intent of 4.1.1](#) is to provide consistency so that different user agents or assistive technologies will yield the same result;
- [4.1.2 Name, Role, Value](#)—requires information in a programmatically determinable form.
- [4.1.3 Status Messages](#)—requires information in a programmatic determinable form. Additionally, software with closed functionality is not typically implemented using markup languages.

#### NOTE 3

Non-web software with closed functionality would need equivalent facilitation to provide access to status messages.

## B. Background on Text / Command-line / Terminal Applications and Interfaces

### B.1 How text interfaces are realized

The interface of a text application is realized through a server application directing which characters should be placed on the screen, along with either a hardware terminal or a terminal application that displays the characters. The client terminal application for text applications is analogous to a web user agent for web pages. Also like web applications, text applications may execute primarily on a remote server or execute locally.

Some text applications render like a TeleTYpewriter (TTY); their output is always appended, like an ever growing file. Such text applications are often called “command-line applications” or occasionally “TTY-applications”, and their output can optionally be redirected to a file for later review. Others explicitly place text into a matrix of fixed width character cells on a screen (sometimes with specific foreground and background colors).

Historically, input to the text application itself is provided exclusively through a keyboard interface, though Automatic Speech Recognition (ASR) based voice input is sometimes now an alternative option - especially on mobile devices.

## B.2 How text applications have been made accessible via assistive technology

Strategies for making text applications accessible through assistive technology involve two key tasks: (1) obtaining all of the text displayed in the interface, and (2) performing an analysis on that text to detect screen updates and attempt to discern structural elements.

For example, a text application screen reader might directly access the matrix of character cells in the interface and provide a screen review mechanism for the user to review that matrix of characters (by sending the output to synthetic speech and/or a braille display). Alternately, a text application screen reader might directly consume the output rendered (perhaps by acting as its own terminal application or by analyzing the “TTY” output). A text application screen reader might also attempt to analyze the spacing and layout of the text in the matrix, to provide features such as reading columns of text in a multi-column layout; discerning headers through analysis of line spacing, indentation, and capitalization; and discerning input fields or user interface components by scanning for the use of inverse video, for text appearing in brackets, or for text from the character graphics codepage (ASCII codes greater than ‘0x7F’). Some of this analysis might also be done through the use of filter tools that transform the output of a program (e.g., through reformatting “TTY” output rendered to a file or as direct input to a filter tool).

Similarly, a text application screen magnifier would gain access to the matrix of character cells to magnify them or re-display them in a larger font. It would scan for screen refreshes and updates and then apply heuristics to what had changed in order to decide what sub-matrix of character cells should appear in a magnified view. It would also scan for inverse video and a moving text cursor to track text being input by the user (and might combine the text matrix scanning with scanning of the keyboard input to match user input to what is appearing on the screen).

## § B.3 Applying WCAG 2.2 to text applications

To apply WCAG to text applications, it is necessary to apply the glossary terms [accessibility supported](#) and [programmatically determined](#) in the context of how text applications are rendered and the history of assistive technologies that made them accessible.

As noted above, in a text interface the terminal application renders the characters on the screen, just as a Web browser typically renders content for a Web application. As an example, for success criterion [1.4.4 Resize Text](#), a text application could achieve 200 percent resizing when the terminal application client that is rendering it has this capability (cf. WCAG 2.2 Technique [G142 Using a technology that has commonly-available user agents that support zoom](#)). Many web pages and web applications use this approach to meet success criterion [1.4.4 Resize Text](#) through no explicit action of their own.

A similar approach could also be used for success criterion [1.4.3 Contrast \(minimum\)](#) (cf. WCAG 2.2 Technique [G148: Not specifying background color, not specifying text color, and not using technology features that change those defaults](#)): relying on the terminal application client to render the text with sufficient contrast against the background. In fact, many terminal applications allow the user to force all text to share a single user-chosen foreground color (and a single user-chosen background color), overriding the text application's specified colors to meet the user's desires or needs.

Since many assistive technology analysis techniques depend upon discerning the location of the text input cursor, terminal application use of “soft cursors” and “highlight bars” may bypass those analysis techniques and cause failures of success criteria.

### NOTE 1

It is outside of the scope of this document to define WCAG techniques for non-web ICT. These examples are simply illustrations of how WCAG 2.2 success criteria can be applied to this class of non-web software applications.

The way to think about “accessibility supported” and “programmatically determined” may seem a little different for text applications, but the definitions are unchanged. Unlike the semantic objects of graphical user interfaces and web pages, the output of text-based applications consists of plain text. A terminal emulator acts as the user agent for text-based applications; it might render some content such as escape codes as semantic elements, but otherwise exposes only lines of text to assistive technology. Where assistive

technology is able to interpret the text and any semantic objects accurately, the content is "programmatically determinable"—even though no explicit markup was necessarily used to make it so.

#### NOTE 2

The terminal application itself is “traditional” non-web software ICT. It is only for the text application that there is a need to take this approach with these glossary terms.

## § C. Acknowledgements

### § C.1 Participants of the WCAG2ICT Task Force Active in the Development of this Document

- Shadi Abou-Zahra (Amazon)
- Charles Adams (Oracle Corporation)
- Bruce Bailey (U.S. Access Board)
- Fernanda Bonnin (Microsoft)
- Devanshu Chandra (Deque Systems, Inc.)
- Michael Cooper (W3C)
- Phil Day (NCR)
- Mitchell Evan (TPGi)
- Olivia Hogan-Stark (NCR)
- Thorsten Katzmann (IBM)
- Chris Loiselle (Oracle Corporation)
- Loïc Martínez Normand (Universidad Politécnica de Madrid)
- Laura Miller (TPGi)
- Daniel Montalvo (W3C)
- Mary Jo Mueller (IBM)
- Sam Ogami (Invited expert)

- Mike Pluke (Invited expert)
- Shawn Thompson (Shared Services Canada)
- Bryan Trogon (Google LLC)
- Gregg Vanderheiden (Raising the Floor)

## § C.2 Participants in the AG Working Group that Actively Reviewed and Contributed

### EDITOR'S NOTE

This list will be updated as AG WG reviews and contributions are completed.

- Jonathan Avila (Level Access)
- Daniel Bjorge (Microsoft)
- Alastair Campbell (Nomensa)
- Laura Carlson (Invited expert)
- Jennifer Delisi (Invited expert)
- Detlev Fischer (Invited expert)
- Jan Jaap de Groot (Invited expert)
- Andrew Kirkpatrick (Adobe)
- Patrick Lauke (TetraLogical)
- Todd Libby (Invited expert)
- David MacDonald (Invited expert)
- Rachael Bradley Montgomery (Library of Congress)
- Kimberly Patch (Invited expert)

## § C.3 Participants in the APA Working Group that Contributed

Special thanks goes to members of the APA working group that contributed their expertise to updates in the Text / Command-line / Terminal Applications and Interfaces content.

- Janina Sajka (Invited expert)
- Matthew Atkinson (TPGi)
- Jason White (Invited expert)

## § C.4 Previous Contributors

The following people contributed to the development of the 2013 WCAG2ICT Note.

Shadi Abou-Zahra, Bruce Bailey, Judy Brewer, Michael Cooper, Pierce Crowell, Allen Hoffman, Kiran Kaja, Andrew Kirkpatrick, Peter Korn, Alex Li, David MacDonald, Mary Jo Mueller, Loïc Martínez Normand, Mike Pluke, Janina Sajka, Andi Snow-Weaver, Gregg Vanderheiden

## § C.5 Enabling Funders

This publication has been funded in part by funds from the following organizations:

- Ford Foundation
- European Commission

The content of this publication does not necessarily reflect the views or policies of the Ford Foundation and/or the European Commission, nor does mention of trade names, commercial products, or organizations imply endorsement by the aforementioned organizations.

## § D. References

### § D.1 Informative references

#### **[UNDERSTANDING-WCAG22]**

*Understanding Web Content Accessibility Guidelines 2.2.* URL: <https://www.w3.org>

[/WAI/WCAG22/Understanding/](#)

**[WCAG22]**

[Web Content Accessibility Guidelines \(WCAG\) 2.2](#). Michael Cooper; Andrew Kirkpatrick; Alastair Campbell; Rachael Bradley Montgomery; Charles Adams. W3C. 20 July 2023. W3C Proposed Recommendation. URL: <https://www.w3.org/TR/WCAG22/>

**[WCAG22-TECHS]**

[Techniques for WCAG 2.2](#). URL: <https://www.w3.org/WAI/WCAG22/Techniques/>

[↑](#)