W3C Recommendation

# Accessible Name and Description Computation 1.1

## W3C Recommendation 18 December 2018

**This version:**
https://www.w3.org/TR/2018/REC-accname-1.1-20181218/

**Latest published version:**
https://www.w3.org/TR/accname-1.1/

**Latest editor's draft:**
https://w3c.github.io/accname/

**Implementation report:**
https://w3c.github.io/test-results/accname/

**Previous version:**
https://www.w3.org/TR/2018/PR-accname-1.1-20181018/

**Previous Recommendation:**
https://www.w3.org/TR/wai-aria-implementation-1.0/

**Editors:**
Joanmarie Diggs (Igalia, S.L.)
Bryan Garaventa (Level Access)
Michael Cooper (W3C)

**Former editors:**
Richard Schwerdtfeger (Knowbility) (Editor until October 2017)
Joseph Scheuhammer (Inclusive Design Research Centre, OCAD University) (Editor until May 2017)
James Craig (Apple Inc.) (Editor until May 2016)
Andi Snow-Weaver (IBM) (Editor until December 2012)
Aaron Leventhal (IBM) (Editor until January 2009)

Please check the **errata** for any errors or issues reported since publication.

See also **translations**.



## Abstract

This document describes how *user agents* determine the *names* and *descriptions* of *accessible objects* from web content languages. This information is in turn exposed through *accessibility APIs* so that *assistive technologies* can identify these objects and present their names or descriptions to users.

Documenting the algorithm through which names and descriptions are to be determined promotes interoperable exposure of these properties among different accessibility APIs and helps to ensure that this information appears in a manner consistent with author intent.

The accessible name and description computation specification defines support that applies across multiple content technologies. This includes accessible name and description provided by general-purpose *WAI-ARIA* [WAI-ARIA] *roles*, *states*, and *properties* as well as features specific to individual content languages.

This document supersedes the accessible name and description guidance in the WAI-ARIA 1.0 User Agent Implementation Guide [WAI-ARIA-IMPLEMENTATION] W3C Recommendation. It is part of the WAI-ARIA suite described in the WAI-ARIA Overview.

## Status of This Document

*This section describes the status of this document at the time of its publication. Other documents may supersede this document. 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 the Accessible Name and Description Computation (Accname) 1.1 W3C Recommendation by the Accessible Rich Internet Applications Working Group. The Working Group created a Accname 1.1 Implementation Report to demonstrate that the specification is implementable. A history of changes to Accname 1.1 is available in the appendix.

To comment on this document, file an issue in the W3C accname GitHub repository. If this is not feasible, send email to public-aria@w3.org (comment archive). Comments received on the Accname 1.1 Recommendation cannot result in changes to this version of the specification, but may be addressed in errata or future versions of Accname. The Working Group may not make formal responses to comments but future work undertaken by the Working Group may address comments received on this document. In-progress updates to the technology may be viewed in the publicly visible editors' draft.

This document was published by the Accessible Rich Internet Applications Working Group as a Recommendation.

Please see the Working Group's implementation report.

This document has been reviewed by W3C Members, by software developers, and by other W3C groups and interested parties, and is endorsed by the Director as a W3C Recommendation. It is a stable document and may be used as reference material or cited from another document. W3C's role in making the Recommendation is to draw attention to the specification and to promote its widespread deployment. This enhances the functionality and interoperability of the Web.

This document was produced by a group operating under the W3C Patent Policy. W3C maintains a public list of any patent disclosures made in connection with the deliverables of the group; that page also includes instructions for disclosing a patent. An individual who has actual knowledge of a patent which the individual believes contains Essential Claim(s) must disclose the information in accordance with section 6 of the W3C Patent Policy.

This document is governed by the 1 February 2018 W3C Process Document.

## Table of Contents

# 1. Introduction §

*This section is non-normative.*

*User agents* acquire information from the DOM [DOM] and create a parallel structure called the *accessibility tree*, made up of *accessible objects*. An accessible object provides information about its *role*, *states*, and *properties*. An example is an accessible object whose role is `menuitem`, is currently in an `enabled` state, with a `haspopup` property, indicating that it leads to a sub-menu.

The two properties of accessible objects described in this document are its *accessible name* and *accessible description*. The name is a short label that provides information about the purpose of the object. An example of an accessible name for a menu item is `New`, signifying that the menu item provides for the creation of new documents, windows, and so on.

The description is a short explanation that further clarifies the nature of the accessible object. It is not always necessary to provide a description if the name is sufficient, but it can help a user better understand the use of the object.

*Accessibility APIs* currently support flat, unstructured strings for accessible names and descriptions. The result of the name/description computation is thus a flat string.

The terms "accessible name" and "accessible description" are used to emphasize that they are properties of *accessible objects* as exposed by *Accessibility APIs*. However, they are frequently referred to hereafter as simply "name" and "description".

# 2. Conformance §

As well as sections marked as non-normative, all authoring guidelines, diagrams, examples, and notes in this specification are non-normative. Everything else in this specification is normative.

The key word ***MUST*** is to be interpreted as described in [RFC2119].

## 2.1 RFC-2119 Keywords §

RFC-2119 keywords are formatted in uppercase and contained in a `strong` element with `class="rfc2119"`. When the keywords shown above are used, but do not share this format, they do not convey formal information in the RFC 2119 sense, and are merely explanatory, i.e., informative. As much as possible, such usages are avoided in this specification.

## 2.2 Normative and Informative Sections §

The indication whether a section is *normative* or *non-normative* (informative) applies to the entire section including sub-sections.

Informative sections provide information useful to understanding the specification. Such sections may contain examples of recommended practice, but it is not required to follow such recommendations in order to conform to this specification.

## 3. Important Terms §

While some terms are defined in place, the following definitions are used throughout this document.

***Accessibility API***

Operating systems and other platforms provide a set of interfaces that expose information about *objects* and *events* to assistive technologies. Assistive technologies use these interfaces to get information about and interact with those *widgets*. Examples of accessibility APIs are Microsoft Active Accessibility [MSAA], Microsoft User Interface Automation [UI-AUTOMATION], MSAA with *UIA Express* [UIA-EXPRESS], the Mac OS X Accessibility Protocol [AXAPI], the *Linux/Unix Accessibility Toolkit* [ATK] and *Assistive Technology Service Provider Interface* [AT-SPI], and IAccessible2 [IAccessible2].

***Accessibility Tree***

Tree of *accessible objects* that represents the structure of the user interface (UI). Each node in the accessibility tree represents an element in the UI as exposed through the accessibility API; for example, a push button, a check box, or container.

***Accessible Description***

An accessible description provides additional information, related to an interface element, that complements the accessible name. The accessible description might or might not be visually perceivable.

***Accessible Name***

The accessible name is the name of a user interface element. Each platform accessibility API provides the accessible name property. The value of the accessible name may be derived from a visible (e.g., the visible text on a button) or invisible (e.g., the text alternative that describes an icon) property of the user interface element. See related accessible description.

A simple use for the accessible name property may be illustrated by an "OK" button. The text "OK" is the accessible name. When the button receives focus, assistive technologies may concatenate the platform's role description with the accessible name. For example, a screen reader may speak "push-button OK" or "OK button". The order of concatenation and specifics of the role description (e.g., "button", "push-button", "clickable button") are determined by platform *accessibility APIs* or assistive technologies.

***Accessible object***

A node in the accessibility tree of a platform accessibility API. Accessible objects expose various *states*, *properties*, and *events* for use by assistive technologies. In the context of markup languages (e.g., HTML and SVG) in general, and of WAI-ARIA in particular, markup *elements* and their *attributes* are represented as accessible objects.

***Assistive Technologies***

Hardware and/or software that:

- relies on services provided by a user agent to retrieve and render Web content
- works with a user agent or web content itself through the use of APIs, and
- provides services beyond those offered by the user agent to facilitate user interaction with web content by people with disabilities

This definition may differ from that used in other documents.

Examples of assistive technologies that are important in the context of this document include the following:

- screen magnifiers, which are used to enlarge and improve the visual readability of rendered text and images;
- screen readers, which are most-often used to convey information through synthesized speech or a refreshable Braille display;
- text-to-speech software, which is used to convert text into synthetic speech;
- speech recognition software, which is used to allow spoken control and dictation;
- alternate input technologies (including head pointers, on-screen keyboards, single switches, and sip/puff devices), which are used to simulate the keyboard;
- alternate pointing devices, which are used to simulate mouse pointing and clicking.

***Attribute***

In this specification, attribute is used as it is in markup languages. Attributes are structural features added to *elements* to provide information about the *states* and *properties* of the *object* represented by the element.

***Class***

A set of instance *objects* that share similar characteristics.

***Element***

In this specification, element is used as it is in markup languages. Elements are the structural elements in markup language that contains the data profile for *objects*.

***Event***

A programmatic message used to communicate discrete changes in the state of an object to other objects in a computational system. User input to a web page is commonly mediated through abstract events that describe the interaction and can provide notice of changes to the state of a document object. In some programming languages, events are more commonly known as notifications.

***Hidden***

Indicates that the element is not visible, perceivable, or interactive to *any* user. An element is considered *hidden* if it or any one of its ancestor elements is not rendered or is explicitly hidden.

***Informative***

Content provided for information purposes and not required for conformance. Content required for conformance is referred to as normative.

***Node***

Basic type of *object* in the DOM tree or *accessibility tree*. DOM nodes are further specified as *Element* or *Text nodes*, among other types. The nodes of an *accessibility tree* are *accessible objects*.

***Normative***

Required for conformance. By contrast, content identified as informative or "non-normative" is not required for conformance.

***Object***

In the context of user interfaces, an item in the perceptual user experience, represented in markup languages by one or more *elements*, and rendered by *user agents*.

In the context of programming, the instantiation of one or more *classes* and interfaces which define the general characteristics of similar objects. An object in an accessibility API may represent one or more DOM objects. *Accessibility APIs* have defined interfaces that are distinct from DOM interfaces.

***Perceivable***

Presentable to users in ways they can sense. References in this document relate to *WCAG 2.1 Principle 1: Content must be perceivable* [WCAG21].

***Property***

*Attributes* that are essential to the nature of a given object, or that represent a data value associated with the object. A change of a property may significantly impact the meaning or presentation of an object. Certain properties (for example, `aria-multiline`) are less likely to change than *states*, but note that the frequency of change difference is not a rule. A few properties, such as `aria-activedescendant`, `aria-valuenow`, and `aria-valuetext` are expected to change often. See clarification of states versus properties.

***Role***

Main indicator of type. This *semantic* association allows tools to present and support interaction with the object in a manner that is consistent with user expectations about other objects of that type.

***Semantics***

The meaning of something as understood by a human, defined in a way that computers can process a representation of an object, such as *elements* and *attributes*, and reliably represent the object in a way that various humans will achieve a mutually consistent understanding of the object.

***State***

A state is a dynamic *property* expressing characteristics of an object that may change in response to user action or automated processes. States do not affect the essential nature of the object, but represent data associated with the object or user interaction possibilities. See clarification of states versus properties.

***Text node***

Type of DOM *node* that represents the textual content of an *attribute* or an *element*. A Text node has no child nodes.

***Tooltip attribute***
: Any host language attribute that would result in a user agent generating a tooltip such as in response to a mouse hover in desktop user agents.

***User Agent***
: Any software that retrieves, renders and facilitates end user interaction with Web content. This definition may differ from that used in other documents.

***Widget***
: Discrete user interface *object* with which the user can interact. Widgets range from simple objects that have one value or operation (e.g., check boxes and menu items), to complex objects that contain many managed sub-objects (e.g., trees and grids).

## 4. Name and Description §

The starting point of the name and description computation is a DOM *element*. The output is a flat, unstructured string that can be as simple as a single word, or a string of space-separated tokens. Examples include `Save` and `Reload from disk`.

An important factor is the *element*'s *role*, that determines which content contributes to the name string. Roles have a `nameFrom` RDF property, with two possible values:

**author**
: name is generated from values provided by the author in explicit markup features such as the `aria-label` and `aria-labelledby` *attribute*, or a host language labeling mechanism, such as the `alt` or `title` *attribute* in HTML, or the `desc` *element* in SVG.

**contents**
: name is generated from the *Text nodes* associated with the *element*. Although this may be allowed in addition to "author" in some *roles*, "content" is used only if higher priority "author" features are not provided. Priority is defined by the accessible name and description computation algorithm.

The *Accessible Rich Internet Applications (WAI-ARIA) 1.1* [WAI-ARIA] specification provides lists of roles that support name from author and name from content.

### 4.1 Name Computation §

*User agents* ***MUST*** compute an *accessible name* using the rules outlined below in the section titled Accessible Name and Description Computation.

## 4.2 Description Computation §

If `aria-describedby` is present, *user agents* ***MUST*** compute the accessible description by concatenating the text alternatives for elements referenced by an `aria-describedby` attribute on the current element. The text alternatives for the referenced elements are computed using a number of methods, outlined below in the section titled Accessible Name and Description Computation.

## 4.3 Accessible Name and Description Computation §

The accessible name and description computation is used to generate both the *accessible name* and *accessible description*. There are different rules provided for several different types of *elements*, *nodes*, and combinations of markup. Text alternatives are built up, when appropriate, from all the relevant content contained within an *element*. This is accomplished via steps 2B and 2F, which are recursive, using the full set of rules to retrieve text from its own children or nodes it references.

The purpose of the computation is to create a *perceivable* label or description for alternative presentations, in the form of a flat string of space separated textual tokens.

### 4.3.1 Terminology §

**Root node**
: The DOM *node* or *element* for which the text alternative is sought.

**Current node**
: The DOM *node* currently traversed to compute the `root node`'s text equivalent. Initially, the `current node` is the `root node`, but at later stages is either some descendant of the `root node`, or another referenced node.

**Flat string**
: A string of characters where all carriage returns, newlines, tabs, and form-feeds are replaced with a single space, and multiple spaces are reduced to a single space. The string contains only character data; it does not contain any markup.

**Total accumulated text**
: The text equivalent computed up to, but not including the `current node`.

**Accumulated text**
: Text accumulated at a step or sequence of steps described below. It is temporary storage for those steps.

**Result**
: The text equivalent computed at one of the steps described below.

**Append the result, without a space, to X**

- If X is empty, copy the `result` to X.
- If X is non-empty, copy the `result` to the end of X.

**Append the result, with a space, to X**

- If X is empty, copy the `result` to X.
- If X is non-empty, add a space to the end of X and then copy the `result` to X after the space.

**Prepend result, without a space, to X**

- If X is empty, copy the `result` to X.
- If X is non-empty, copy the `result` to the start of X.

**Prepend the result, with a space, to X**

- If X is empty, copy the `result` to X.
- If X is non-empty, copy the `result` to the start of X, and add a space after the copy.

The text alternative for a given element is computed as follows:

1. Initialize: Set the `root node` to the given element, the `current node` to the `root node`, and the `total accumulated text` to the empty string ("").
2. Compute the text alternative for the `current node`:
   A. If the `current node` is *hidden* **and is not** directly referenced by `aria-labelledby` or `aria-describedby`, nor directly referenced by a native host language text alternative *element* (e.g. `label` in HTML) or *attribute*, return the empty string.
   ▶ **Comment:**
   B. Otherwise:
   - if computing a *name*, and the `current node` has an `aria-labelledby` *attribute* that contains at least one valid IDREF, and the `current node` is not already part of an `aria-labelledby` traversal, process its IDREFs in the order they occur:
   - or, if computing a *description*, and the `current node` has an `aria-describedby` *attribute* that contains at least one valid IDREF, and the `current node` is not already part of an `aria-describedby` traversal, process its IDREFs in the order they occur:
     i. Set the `accumulated text` to the empty string.
     ii. For each IDREF:
       a. Set the `current node` to the node referenced by the IDREF.
       b. Compute the text alternative of the `current node` beginning with step 2. Set the `result` to that text alternative.
       c. Append the `result`, with a space, to the `accumulated text`.
     iii. Return the `accumulated text`.
   ▶ **Example:**
   C. Otherwise, if computing a *name*, and if the `current node` has an `aria-label` *attribute* whose value is not the empty string, nor, when trimmed of white space, is not the empty string:
   - If traversal of the `current node` is due to recursion **and** the `current node` is an embedded control as defined in step 2E, ignore `aria-label` and skip to rule 2E.

- Otherwise, return the value of `aria-label`.

▶ **Example:**

D. Otherwise, if the `current node`'s native markup provides an *attribute* (e.g. `title`) or *element* (e.g. HTML `label`) that defines a text alternative, return that alternative in the form of a `flat string` as defined by the host language, unless the element is marked as presentational (`role="presentation"` or `role="none"`).

▶ **Comment:**

E. Otherwise, if the `current node` is a control embedded within the label (e.g. the `label` element in HTML or any element directly referenced by `aria-labelledby`) for another *widget*, where the user can adjust the embedded control's value, then include the embedded control as part of the text alternative in the following manner:

- If the embedded control has role textbox, return its value.
- If the embedded control has role menu button, return the text alternative of the button.
- If the embedded control has role combobox or listbox, return the text alternative of the chosen option.
- If the embedded control has role range (e.g., a spinbutton or slider):
  - If the `aria-valuetext` property is present, return its value,
  - Otherwise, if the `aria-valuenow` property is present, return its value,
  - Otherwise, use the value as specified by a host language attribute.

▶ **Example:**

F. Otherwise, if the `current node's` *role* allows name from content, or if the `current node` is referenced by `aria-labelledby`, `aria-describedby`, or is a native host language text alternative *element* (e.g. `label` in HTML), or is a descendant of a native host language text alternative *element*:

i. Set the `accumulated text` to the empty string.

ii. Check for CSS generated textual content associated with the `current node` and include it in the `accumulated text`. The CSS `:before` and `:after` pseudo elements [CSS2] can provide textual content for *elements* that have a content model.

- For `:before` pseudo elements, *User agents* ***MUST*** prepend CSS textual content, without a space, to the textual content of the `current node`.
- For `:after` pseudo elements, *User agents* ***MUST*** append CSS textual content, without a space, to the textual content of the `current node`.

iii. For each child node of the `current node`:

a. Set the `current node` to the child node.

b. Compute the text alternative of the `current node` beginning with step 2. Set the `result` to that text alternative.

c. Append the `result` to the `accumulated text`.

iv. Return the `accumulated text`.

**Important**: Each *node* in the subtree is consulted only once. If text has been collected from a descendant, but is referenced by another IDREF in some descendant node, then that second, or subsequent, reference is not followed. This is done to avoid infinite loops.

▶ **Comment:**

G. Otherwise, if the `current node` is a *Text node*, return its textual contents.

H. Otherwise, if the `current node` is a descendant of an element whose *Accessible Name* or *Accessible Description* is being computed, and contains descendants, proceed to 2F.i.

I. Otherwise, if the `current node` has a *Tooltip attribute*, return its value.

▶ **Comment:**

Append the `result` of each step above, with a space, to the `total accumulated text`.

After all steps are completed, the `total accumulated text` is used as the *accessible name* or *accessible description* of the *element* that initiated the computation.

# 5. Accessible Name and Description Mapping §

Information concerning name and description accessibility API mappings, including relationships, such as labelled-by/label-for and described-by/description-for, is documented in the Core Accessibility API Mappings specification [CORE-AAM-1.1]. See the mapping table entries for `aria-label`, `aria-labelledby`, and `aria-describedby`.

# 6. Appendices §

## 6.1 Change Log §

### 6.1.1 Substantive changes since the last public working draft §

- 10-April-2018: Treat listbox in the same fashion as combobox in step 2E, i.e. returning the text alternative of the chosen option.
- 5-March-2018: Remove mapping table content as it is redundant to what is in Core AAM.
- 9-August-2017: Added interim step to be able to process recursions of elements who are descendants of an element whose name/description is being computed
- 27-July-2017: Modify step 2F to handle elements that are descendants of text alternative elements in host languages.
- 08-June-2017: Remove list style information from the accessible name computation.

#### 6.1.2 Other substantive changes since the WAI-ARIA 1.0 User Agent Implementation Guide Recommendation §

- 04-Apr-2016: Added new UIA FullDescription property for accessible description mapping.
- 05-Nov-2015: Removed all MSAA+UIA Express mappings.
- 10-Jun-2015: Moved special case of unlabeled `<img>` to HTML-AAM.
- 05-Jan-2015: Glossary entry for "value" removed; removed links to that entry.

## 6.2 Acknowledgments §

*This section is non-normative.*

The following people contributed to the development of this document.

#### 6.2.1 Participants active in the ARIA WG at the time of publication §


- Ann Abbott (Invited Expert)
- Irfan Ali (Educational Testing Service)
- Amelia Bellamy-Royds (Invited Expert)
- Zoë Bijl (Invited Expert)
- David Bolter (Mozilla Foundation)
- Bogdan Brinza (Microsoft Corporation)
- Shari Butler (Pearson plc)
- Thaddeus Cambron (Invited Expert)
- Michael Cooper (W3C Staff)
- James Craig (Apple Inc.)
- Joanmarie Diggs (Igalia)
- John Foliot (Deque Systems, Inc.)
- Bryan Garaventa (SSB BART Group)
- Matt Garrish (DAISY Consortium)
- Becky Gibson (Invited Expert)
- Glen Gordon (The Paciello Group, LLC)
- Jon Gunderson (University of Illinois at Urbana-Champaign)
- Matthew King (Facebook)
- JaEun Jemma Ku (University of Illinois at Urbana-Champaign)
- Charles LaPierre (Benetech)

- Aaron Leventhal (Google, Inc.)
- Dominic Mazzoni (Google, Inc.)
- Shane McCarron (Invited Expert, Aptest)
- Jan McSorley (Pearson plc)
- James Nurthen (Oracle Corporation)
- Ian Pouncey (The Paciello Group, LLC)
- Ruoxi Ran (W3C Staff)
- Janina Sajka (Invited Expert, The Linux Foundation)
- Stefan Schnabel (SAP SE)
- Lisa Seeman-Kestenbaum (Invited Expert)
- Tzviya Siegman (Wiley)
- Alexander Surkov (Mozilla Foundation)
- Job van Achterberg (Invited Expert)
- Evan Yamanishi (W. W. Norton)
- Jason White (Educational Testing Service)

### 6.2.2 Other ARIA contributors, commenters, and previously active participants §

- Shadi Abou-Zahra (W3C)
- Jim Allan (TSB)
- Jonny Axelsson (Opera Software)
- David Baron (Mozilla Foundation)
- Art Barstow (Nokia Corporation)
- Simon Bates
- Christy Blew (University of Illinois at Urbana-Champaign)
- Chris Blouch (AOL)
- Judy Brewer (W3C/MIT)
- Mark Birbeck (Sidewinder Labs)
- Sally Cain (Royal National Institute of Blind People (RNIB))
- Gerardo Capiel (Benetech)
- Ben Caldwell (Trace)
- Sofia Celic-Li
- Jaesik Chang (Samsung Electronics Co., Ltd.)
- Alex Qiang Chen (University of Manchester)
- Charles Chen (Google, Inc.)

- Christian Cohrs
- Deborah Dahl
- Erik Dahlström (Opera Software)
- Dimitar Denev (Frauenhofer Gesellschaft)
- Micah Dubinko (Invited Expert)
- Mandana Eibegger
- Beth Epperson (Websense)
- Fred Esch (IBM Corporation)
- Donald Evans (AOL)
- Steve Faulkner (The Paciello Group, LLC)
- Chris Fleizach (Apple Inc.)
- Kelly Ford (Microsoft Corporation)
- Geoff Freed (Invited Expert, NCAM)
- Christopher Gallelo (Microsoft Corporation)
- Billy Gregory (The Paciello Group, LLC)
- Karl Groves (The Paciello Group, LLC)
- Birkir Gunnarsson (Deque Systems, Inc.)
- Kentarou Fukuda (IBM Corporation)
- Bryan Garaventa
- Guido Geloso
- Ali Ghassemi
- Alfred S. Gilman
- Andres Gonzalez (Adobe Systems Inc.)
- Scott González (JQuery Foundation)
- James Graham
- Georgios Grigoriadis (SAP AG)
- Jeff Grimes (Oracle)
- Loretta Guarino Reid (Google, Inc.)
- Markus Gylling (DAISY Consortium)
- Markku Hakkinen (Educational Testing Service)
- Katie Haritos-Shea (Knowbility)
- Barbara Hartel
- James Hawkins (Google, Inc.)
- Benjamin Hawkes-Lewis

- Sean Hayes (Microsoft Corporation)
- Mona Heath (University of Illinois at Urbana-Champaign)
- Jan Heck
- Shawn Henry
- Tina Homboe
- Nicholas Hoyt
- John Hrvatin (Microsoft Corporation)
- Takahiro Inada
- Masayasu Ishikawa (W3C)
- Jim Jewitt
- Kenny Johar (Microsoft Corporation)
- Shilpi Kapoor (BarrierBreak Technologies)
- Masahiko Kaneko (Microsoft Corporation)
- Marjolein Katsma
- Susann Keohane (IBM Corporation)
- George Kerscher (International Digital Publishing Forum)
- Jason Kiss (Department of Internal Affairs, New Zealand Government)
- Todd Kloots
- Jamie Knight (British Broadcasting Corporation)
- Johannes Koch
- Gerard K. Cohen
- Sam Kuper
- Earl Johnson (Sun)
- Jael Kurz
- Rajesh Lal (Nokia Corporation)
- Diego La Monica (International Webmasters Association / HTML Writers Guild (IWA-HWG))
- Gez Lemon (International Webmasters Association / HTML Writers Guild (IWA-HWG))
- Alex Li (SAP)
- Chris Lilley
- Thomas Logan (HiSoftware Inc.)
- Brian Loh
- William Loughborough (Invited Expert)
- Linda Mao (Microsoft)
- David MacDonald (Invited Expert, CanAdapt Solutions Inc.)

- Carolyn MacLeod
- Anders Markussen (Opera Software)
- Krzysztof Maczyński
- Matthew May (Adobe Systems Inc.)
- Mark McCarthy
- Charles McCathie Nevile (Yandex)
- Heather Migliorisi (Invited Expert)
- Mary Jo Mueller (IBM Corporation)
- Alexandre Morgaut (4D)
- Ann Navarro (Invited Expert)
- Joshue O Connor (Invited Expert, CFIT)
- Artur Ortega (Microsoft Corporation)
- Sailesh Panchang (Deque)
- Lisa Pappas (Society for Technical Communication (STC))
- Marta Pawlowlska (Samsung Electronics Co., Ltd.)
- Dave Pawson (RNIB)
- Steven Pemberton (CWI Amsterdam)
- Simon Pieters (Opera Software)
- Jean-Bernard Piot (4D)
- David Poehlman, Simon Pieters (Opera Software)
- Sarah Pulis (Media Access Australia)
- T.V. Raman (Google, Inc.)
- Jan Richards
- Gregory Rosmaita (Invited Expert)
- Tony Ross (Microsoft Corporation)
- Alex Russell (Dojo Foundation) (
- Mark Sadecki (Invited Expert)
- Mario Sánchez Prada (Samsung Electronics Co., Ltd. and Gnome Foundation)
- Martin Schaus (SAP AG)
- Doug Schepers (W3C)
- Cynthia Shelly (Microsoft Corporation)
- Joseph Scheuhammer (Invited Expert, Inclusive Design Research Centre, OCAD University)
- Matthias Schmitt
- Richard Schwerdtfeger (IBM, Knowbility)

- Marc Silbey (Microsoft Corporation)
- Leif Halvard Sili
- Henri Sivonen (Mozilla)
- Michael Smith (W3C)
- Andi Snow-Weaver (IBM Corporation)
- Ville Skyttä
- Henny Swan (BBC)
- Neil Soiffer (Design Science)
- Vitaly Sourikov
- Mike Squillace (IBM)
- Maciej Stachowiak (Apple Inc.)
- Christophe Strobbe
- Suzanne Taylor (Pearson plc)
- Terrill Thompson
- David Todd
- Gregg Vanderheiden (Invited Expert, Trace)
- Anne van Kesteren
- Léonie Watson (The Paciello Group, LLC)
- Wen He (Tencent)
- Wu Wei (W3C / RITT)
- Ryan Williams (Oracle)
- Tom Wlodkowski
- Sam White (Apple Inc.)
- Marco Zehe (Mozilla Foundation)
- Gottfried Zimmermann (Invited Expert, Access Technologies Group)

### 6.2.3 Enabling funders §

This publication has been funded in part with U.S. Federal funds from the Department of Education, National Institute on Disability, Independent Living, and Rehabilitation Research (NIDILRR), initially under contract number ED-OSE-10-C-0067 and currently under contract number HHSP23301500054C. The content of this publication does not necessarily reflect the views or policies of the U.S. Department of Education, nor does mention of trade names, commercial products, or organizations imply endorsement by the U.S. Government.

# A. References §

## A.1 Normative references §

**[AT-SPI]**
*Assistive Technology Service Provider Interface*. The GNOME Project. URL: https://developer.gnome.org/libatspi/stable/

**[ATK]**
*ATK - Accessibility Toolkit*. The GNOME Project. URL: https://developer.gnome.org/atk/stable/

**[AXAPI]**
*The NSAccessibility Protocol for macOS*. Apple, Inc. URL: https://developer.apple.com/documentation/appkit/nsaccessibility

**[CORE-AAM-1.1]**
*Core Accessibility API Mappings 1.1*. Joanmarie Diggs; Joseph Scheuhammer; Richard Schwerdtfeger; Michael Cooper; Andi Snow-Weaver; Aaron Leventhal. W3C. 14 December 2017. W3C Recommendation. URL: https://www.w3.org/TR/core-aam-1.1/

**[CSS2]**
*Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification*. Bert Bos; Tantek Çelik; Ian Hickson; Håkon Wium Lie et al. W3C. 7 June 2011. W3C Recommendation. URL: https://www.w3.org/TR/CSS2/

**[IAccessible2]**
*IAccessible2*. Linux Foundation. URL: https://www.linuxfoundation.org/collaborate/workgroups/accessibility/iaccessible2

**[MSAA]**
*Microsoft Active Accessibility (MSAA) 2.0*. Microsoft Corporation. URL: https://msdn.microsoft.com/en-us/library/ms697707.aspx

**[RFC2119]**
*Key words for use in RFCs to Indicate Requirement Levels*. S. Bradner. IETF. March 1997. Best Current Practice. URL: https://tools.ietf.org/html/rfc2119

**[UI-AUTOMATION]**
*UI Automation*. Microsoft Corporation. URL: https://msdn.microsoft.com/en-us/library/ee684009%28v=vs.85%29.aspx

**[UIA-EXPRESS]**
*The IAccessibleEx Interface*. Microsoft Corporation. URL: https://msdn.microsoft.com/en-us/library/windows/desktop/dd561898%28v=vs.85%29.aspx

**[WAI-ARIA]**
*Accessible Rich Internet Applications (WAI-ARIA) 1.1*. Joanmarie Diggs; Shane McCarron; Michael Cooper; Richard Schwerdtfeger; James Craig. W3C. 14 December 2017. W3C Recommendation. URL: https://www.w3.org/TR/wai-aria-1.1/

**[WAI-ARIA-IMPLEMENTATION]**
*WAI-ARIA 1.0 User Agent Implementation Guide*. Joseph Scheuhammer; Michael Cooper. W3C. 20 March 2014. W3C Recommendation. URL: https://www.w3.org/TR/wai-aria-implementation/

**[WCAG21]**
*Web Content Accessibility Guidelines (WCAG) 2.1*. Andrew Kirkpatrick; Joshue O Connor; Alastair Campbell; Michael Cooper. W3C. 5 June 2018. W3C Recommendation. URL: https://www.w3.org/TR/WCAG21/

## A.2 Informative references §

**[DOM]**
*DOM Standard*. Anne van Kesteren. WHATWG. Living Standard. URL: https://dom.spec.whatwg.org/

↑