Accessible Rich Internet Applications (WAI-ARIA) 1.1

W3C Working Draft 14 May 2015

This version:
http://www.w3.org/TR/2015/WD-wai-aria-1.1-20150514/
Latest published version:
http://www.w3.org/TR/wai-aria-1.1/
Latest editor's draft:
http://w3c.github.io/aria/aria/aria.html
Previous version:
http://www.w3.org/TR/2014/WD-wai-aria-1.1-20141211/
Latest Recommendation:
http://www.w3.org/TR/2014/REC-wai-aria-20140320/
Editors:
Joanmarie Diggs, Igalia, S.L., jdiggs@igalia.com
James Craig, Apple Inc., jcraig@apple.com
Shane McCarron, Applied Testing and Technology, Inc., shane@aptest.com
Michael Cooper, W3C, cooper@w3.org

Abstract

Accessibility of web content requires semantic information about widgets, structures, and behaviors, in order to allow assistive technologies to convey appropriate information to persons with disabilities. This specification provides an ontology of roles, states, and properties that define accessible user interface elements and can be used to improve the accessibility and interoperability of web content and applications. These semantics are designed to allow an author to properly convey user interface behaviors and structural information to assistive technologies in document-level markup. This version adds features new since WAI-ARIA 1.0 [WAI-ARIA-10] to complete the HTML + ARIA accessibility model. It is expected this will complement [HTML5].

This document 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 http://www.w3.org/TR/.

This is a Working Draft of WAI-ARIA 1.1 by the Protocols & Formats Working Group of the Web Accessibility Initiative. This version of ARIA 1.1 adds properties to support tables, the "aria-current" property to indicate the active item in a container, and new roles for search boxes and switch-type checkboxes. This draft is supported by Core Accessibility API Mappings 1.1 [[!CORE-AAM]. A history of changes to WAI-ARIA 1.1 is available in the appendix.

Feedback on any aspect of the specification is accepted. For this publication, the Protocols and Formats Working Group particularly seeks feedback on the following questions:

- Is the "aria-current" property useful and clear?
- Are the table properties ("aria-colcount", "aria-colindex", "aria-colspan", "aria-rowcount", "aria-rowindex", "aria-rowspan") useful and clear?
- Are other changes to states, and properties appropriate?

To comment, send email to public-pfwg-comments@w3.org (comment archive) or file an issue in W3C Bugzilla. Comments are requested by 19 June 2015. In-progress updates to the document may be viewed in the publicly visible editors' draft.

Publication as a Working Draft does not imply endorsement by the W3C Membership. 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.

This document was produced by a group operating under the 5 February 2004 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 August 2014 W3C Process Document.

Table of Contents

§

1. Introduction §

This section is non-normative.

The goals of this specification include:

- expanding the accessibility information that may be supplied by the author;
- requiring that supporting host languages provide full keyboard support that may be implemented in a device-independent way, for example, by telephones, handheld devices, e-book readers, and televisions;
- improving the accessibility of dynamic content generated by scripts; and
- providing for interoperability with assistive technologies.

WAI-ARIA is a technical specification that provides a framework to improve the accessibility and interoperability of web content and applications. This document is primarily for developers creating custom widgets and other web application components. Please see the WAI-ARIA Overview for links to related documents for other audiences, such as the WAI-ARIA Primer that introduces developers to the accessibility problems that WAI-ARIA is intended to solve, the fundamental concepts, and the technical approach of WAI-ARIA.

This draft currently handles two aspects of roles: user interface functionality and structural relationships. For more information and use cases, see the [WAI-ARIA-PRIMER] for the use of roles in making interactive content accessible.

The role taxonomy is designed in part to support the common roles found in platform accessibility APIs. Reference to roles found in this taxonomy by dynamic web content may be used to support interoperability with assistive technologies.

The schema to support this standard has been designed to be extensible so that custom roles can be created by extending base roles. This allows user agents to support at least the base role, and user agents that support the custom role can provide enhanced access. Note that much of this could be formalized in [XMLSCHEMA-2]. However, being able to define similarities between roles, such as baseConcepts and more descriptive definitions, would not be available in XSD.

- WAI-ARIA Primer [WAI-ARIA-PRIMER], a W3C Working Group Note, introduces developers to the accessibility problems that WAI-ARIA is intended to solve, the fundamental concepts, and the technical approach of WAI-ARIA.
- WAI-ARIA Authoring Practices [WAI-ARIA-PRACTICES], a planned W3C Working Group Note, describes how web content developers can develop accessible rich internet applications using WAI-ARIA. It provides detailed advice and examples directed primarily to web application developers, yet also useful to user agent and developers of assistive technologies.
- WAI-ARIA User Agent Implementation Guide [WAI-ARIA-IMPLEMENTATION], a planned W3C Working Group Note, describes how browsers and other user agents should support WAI-ARIA; specifically, how to expose WAI-ARIA features to platform accessibility APIs.
- WAI-ARIA Roadmap [WAI-ARIA-ROADMAP], planned a W3C Working Group Note, defines the path to make rich web content accessible, including steps already taken, remaining future steps, and a time line.

1.1 Rich Internet Application Accessibility §

The domain of web accessibility defines how to make web content usable by persons with disabilities. Persons with certain types of disabilities use assistive technologies (AT) to interact with content. Assistive technologies can transform the presentation of content into a format more suitable to the user, and can allow the user to interact in different ways. For example, the user may need to, or choose to, interact with a slider widget via arrow keys, instead of dragging and dropping with a mouse. In order to accomplish this effectively, the software needs to understand the semantics of the content. Semantics is the science of meaning; in this case, used to assign roles, states, and properties that apply to user interface and content elements as a human would understand. For instance, if a paragraph is semantically identified as such, assistive technologies can interact with it as a unit separable from the rest of the content, knowing the exact boundaries of that paragraph. An adjustable range slider or collapsible list (a.k.a. a tree widget) are more complex examples, in which various parts of the widget have semantics that need to be properly identified for assistive technologies to support effective interaction.

New technologies often overlook semantics required for accessibility, and new authoring practices often misuse the intended semantics of those technologies. Elements that have one defined meaning in the language are used with a different meaning intended to be understood by the user.

For example, web application developers create collapsible tree widgets in HTML using CSS and JavaScript even though HTML has no semantic `tree` element. To a non-disabled user, it may look and act like a collapsible tree widget, but without appropriate semantics, the tree widget may not be perceivable to, or operable by, a person with a disability because assistive technologies may not recognize the role.

The incorporation of WAI-ARIA is a way for an author to provide proper semantics for custom widgets to make these widgets accessible, usable, and interoperable with assistive technologies. This specification identifies the types of widgets and structures that are commonly recognized by accessibility products, by providing an ontology of corresponding roles that can be attached to content. This allows elements with a given role to be understood as a particular widget or structural type regardless of any semantic inherited from the implementing host language. Roles are a common property of platform accessibility APIs which assistive technologies use to provide the user with effective presentation and interaction.

This role taxonomy includes interaction widgets and elements denoting document structure. The role taxonomy describes inheritance and details the attributes each role supports. Information about mapping of roles to accessibility APIs is provided by the WAI-ARIA User Agent Implementation Guide [WAI-ARIA-IMPLEMENTATION].

Roles are element types and will not change with time or user actions. Role information is used by assistive technologies, through interaction with the user agent, to provide normal processing of the specified element type.

States and properties are used to declare important attributes of an element that affect and describe interaction. They enable the user agent and operating system to properly handle the element even when the attributes are dynamically changed by client-side scripts. For example, alternative input and output technology, such as screen readers and speech dictation software, need to be able to recognize and effectively manipulated and communicate various interaction states (e.g.,

disabled, checked) to the user.

While it is possible for assistive technologies to access these properties directly through the Document Object Model [DOM-Level-2-Core], the preferred mechanism is for the user agent to map the states and properties to the accessibility API of the operating system. See the WAI-ARIA User Agent Implementation Guide [WAI-ARIA-IMPLEMENTATION] for details.

Figure 1.0 illustrates the relationship between user agents (e.g., browsers), accessibility APIs, and assistive technologies. It describes the "contract" provided by the user agent to assistive technologies, which includes typical accessibility information found in the accessibility API for many of our accessible platforms for GUIs (role, state, selection, event notification, relationship information, and descriptions). The DOM, usually HTML, acts as the data model and view in a typical model-view-controller relationship, and JavaScript acts as the controller by manipulating the style and content of the displayed data. The user agent conveys relevant information to the operating system's accessibility API, which can be used by any assistive technologies, such as screen readers.

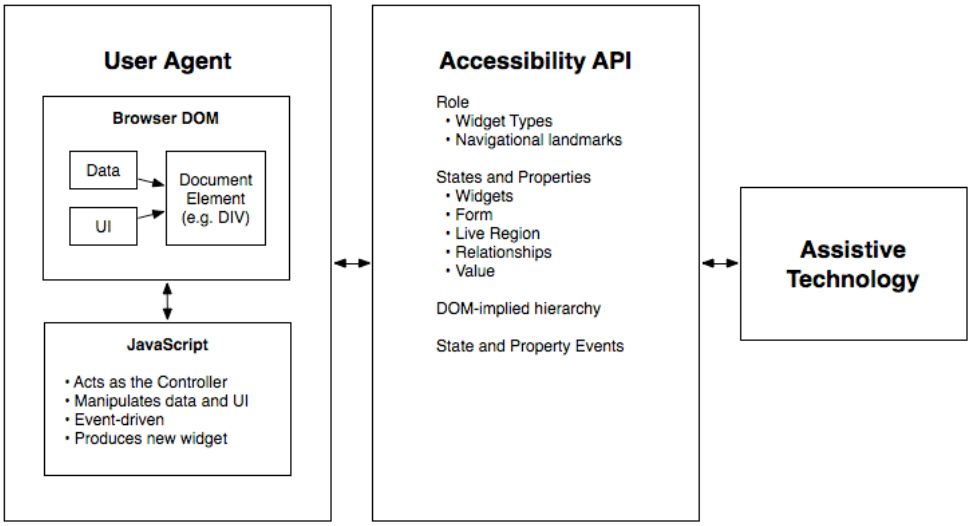


Figure 1: The contract model with accessibility APIs

For more information see the WAI-ARIA Primer [WAI-ARIA-PRIMER] for the use of roles in making interactive content accessible.

In addition to the prose documentation, the role taxonomy is provided in Web Ontology Language (OWL) [OWL-FEATURES], which is expressed in Resource Description Framework (RDF) [RDF-CONCEPTS]. Tools can use these to validate the implementation of roles in a given content document. For example, instances of some roles are expected to be children of a specific parent role. Also, some roles may support a specific state or property that another role does not support.

> NOTE
>
> The use of RDF/OWL as a formal representation of roles may be used to support future extensibility. Standard RDF/OWL mechanisms can be used to define new roles that inherit from the roles defined in this specification. The mechanism to define and use role extensions in an interoperable manner, however, is not defined by this specification. A future version of WAI-ARIA is expected to define how to extend roles.

Users of alternate input devices need keyboard accessible content. The new semantics, when combined with the recommended keyboard interactions provided in WAI-ARIA Authoring Practices [WAI-ARIA-PRACTICES], will allow alternate input solutions to facilitate command and control via an alternate input solution.

WAI-ARIA introduces navigational landmarks through its taxonomy and the XHTML role landmarks, which can help persons with dexterity and vision impairments by providing for improved keyboard navigation. WAI-ARIA may also be used to assist persons with cognitive learning disabilities. The additional semantics allow authors to restructure and substitute alternative content as needed.

Assistive technologies need the ability to support alternative inputs by getting and setting the current value of widget states and properties. Assistive technologies also need to determine what objects are selected and manage widgets that allow multiple selections, such as list boxes and grids.

Speech-based command and control systems can benefit from WAI-ARIA semantics like the `role` attribute to assist in conveying audio information to the user. For example, by determining that an element has a role of `menuitem` each containing text content representing a different flavor, a speech system might state to the user that, "Select one of three choices: chocolate, strawberry, or vanilla."

WAI-ARIA is intended to be used as a supplement for native language semantics, not a replacement. When the host language provides a feature that provides equivalent accessibility to the WAI-ARIA feature, use the host language feature. WAI-ARIA should only be used in cases where the host language lacks the needed role, state, and property indicators. Use a host language feature that is as similar as possible to the WAI-ARIA feature, then refine the meaning by adding WAI-ARIA. For instance, a multi-selectable grid could be implemented as a table, and then WAI-ARIA used to clarify that it is an interactive grid, not just a static data table. This allows for the best possible fallback for user agents that do not support WAI-ARIA and preserves the integrity of the host language semantics.

1.2 Target Audience §

This specification defines the basic model for WAI-ARIA, including roles, states, properties, and values. It impacts

several audiences:

- User agents that process content containing WAI-ARIA features;
- Assistive technologies that present content in special ways to user with disabilities;
- Authors who create content;
- Authoring tools that help authors create conforming content; and
- Conformance checkers that verify appropriate use of WAI-ARIA.

Each conformance requirement indicates the audience to which it applies.

Although this specification is applicable to the above audiences, it is not specifically targeted to, nor is it intended to be the sole source of information for, any of these audiences. The following documents provide important supporting information:

- WAI-ARIA Authoring Practices addresses authoring recommendations, and is also of interest to developers of authoring tools and conformance checkers.
- WAI-ARIA User Agent Implementation Guide addresses developers of user agents and assistive technologies.

1.3 User Agent Support §

WAI-ARIA relies on user agent support for its features in two ways:

- Mainstream user agents use WAI-ARIA to alter how host language features are exposed to accessibility APIs in order to improve accessibility. The mechanism for this is defined in the WAI-ARIA User Agent Implementation Guide [WAI-ARIA-IMPLEMENTATION].
- Assistive technologies use the enhanced information available in an accessibility API, or uses the WAI-ARIA markup directly via the DOM, to convey semantic and interaction information to the user.

Aside from using WAI-ARIA markup to improve what is exposed to accessibility APIs, user agents behave as they would natively. Assistive technologies react to the extra information in the accessibility API as they already do for the same information on non-web content. User agents that are not assistive technologies, however, need do nothing beyond providing appropriate updates to the accessibility API.

The WAI-ARIA specification neither requires nor forbids user agents from enhancing native presentation and interaction behaviors on the basis of WAI-ARIA markup. Mainstream user agents might expose WAI-ARIA navigational landmarks (for example, as a dialog box or through a keyboard command) with the intention to facilitate navigation for all users. User agents are encouraged to maximize their usefulness to users, including users without disabilities.

WAI-ARIA is intended to provide missing semantics so that the intent of the author may be conveyed to assistive technologies. Generally, authors using WAI-ARIA will provide the appropriate presentation and interaction features. Over time, host languages may add WAI-ARIA equivalents, such as new form controls, that are implemented as standard accessible user interface controls by the user agent. This allows authors to use them instead of custom WAI-ARIA enabled user interface components. In this case the user agent would support the native host language feature. Developers of host languages that implement WAI-ARIA are advised to continue supporting WAI-ARIA semantics when they do not adversely conflict with implicit host language semantics, as WAI-ARIA semantics more clearly reflect the intent of the author if the host language features are inadequate to meet the author's needs.

1.4 Co-Evolution of WAI-ARIA and Host Languages §

WAI-ARIA is intended to augment semantics in supporting languages like [HTML5] and [SVG2], or to be used as an accessibility enhancement technology in other markup-based languages that do not explicitly include support for ARIA. It clarifies semantics to assistive technologies when authors create new types of objects, via style and script, that are not yet directly supported by the language of the page, because the invention of new types of objects is faster than standardized support for them appears in web languages.

It is not appropriate to create objects with style and script when the host language provides a semantic element for that type of objects. While WAI-ARIA can improve the accessibility of these objects, accessibility is best provided by allowing the user agent to handle the object natively. For example, it's better to use an `heading` role on a `div` element.

It is expected that, over time, host languages will evolve to provide semantics for objects that currently can only be declared with WAI-ARIA. This is natural and desirable, as one goal of WAI-ARIA is to help stimulate the emergence of more semantic and accessible markup. When native semantics for a given feature become available, it is appropriate for authors to use the native feature and stop using WAI-ARIA for that feature. Legacy content may continue to use WAI-ARIA, however, so the need for user agents to support WAI-ARIA remains.

While specific features of WAI-ARIA may lose importance over time, the general possibility of WAI-ARIA to add semantics to web pages is expected to be a persistent need. Host languages may not implement all the semantics WAI-ARIA provides, and various host languages may implement different subsets of the features. New types of objects are continually being developed, and one goal of WAI-ARIA is to provide a way to make such objects accessible, because web authoring practices often advance faster than host language standards. In this way, WAI-ARIA and host languages both evolve together but at different rates.

Some host languages exist to create semantics for features other than the user interface. For example, SVG expresses the semantics behind production of graphical objects, not of user interface components that those objects may represent; XForms provides semantics for form controls and does not provide wider user interface features. Host languages such as these might, by design, not provide native semantics that map to WAI-ARIA features. In these cases, WAI-ARIA could be adopted as a long-term approach to add semantic information to user interface components.

1.5 Authoring Practices §

1.5.1 Authoring Tools §

Many of the requirements in the definitions of WAI-ARIA roles, state, and properties can be checked automatically during

the development process, similar to other quality control processes used for validating code. To assist authors who are creating custom widgets, authoring tools may compare widget roles, states, and properties to those supported in WAI-ARIA as well as those supported in related and cross-referenced roles, states, and properties. Authoring tools may notify authors of errors in widget design patterns, and may also prompt developers for information that cannot be determined from context alone. For example, a scripting library can determine the labels for the tree items in a tree view, but would need to prompt the author to label the entire tree. To help authors visualize a logical accessibility structure, an authoring environment might provide an outline view of a web resource based on the WAI-ARIA markup.

In HTML, `tabindex` is an important way browsers support keyboard focus navigation for implementations of WAI-ARIA; authoring and debugging tools may check to make sure `tabindex` values are properly set. For example, error conditions may include cases where more than one treeitem in a tree has a `tabindex` value greater than or equal to 0, where `tabindex` is not set on any treeitem, or where `aria-activedescendant` is not defined when the element with the role tree has a `tabindex` value of greater than or equal to 0.

1.5.2 Testing Practices and Tools §

The accessibility of interactive content cannot be confirmed by static checks alone. Developers of interactive content should test for device-independent access to widgets and applications, and should verify accessibility API access to all content and changes during user interaction.

1.6 Assistive Technologies §

Programmatic access to accessibility semantics is essential for assistive technologies. Most assistive technologies interact with user agents, like other applications, through a recognized accessibility API. Perceivable objects in the user interface are exposed to assistive technologies as accessible objects, defined by the accessibility API interfaces. To do this properly, accessibility information - role, states, properties as well as contextual information - needs to be accurately conveyed to the assistive technologies through the accessibility API. When a state change occurs, the user agent provides the appropriate event notification to the accessibility API. Contextual information, in many host languages like HTML, can be determined from the DOM itself as it provides a contextual tree hierarchy.

While some assistive technologies interact with these accessibility APIs, others may access the content directly from the DOM. These technologies can restructure, simplify, style, or reflow the content to help a different set of users. Common use cases for these types of adaptations may be the aging population, persons with cognitive impairments, or persons in environments that interfere with use of their tools. For example, the availability of regional navigational landmarks may allow for a mobile device adaptation that shows only portions of the content at any one time based on its semantics. This could reduce the amount of information the user needed to process at any one time. In other situations it may be appropriate to replace a custom user interface control with something that is easier to navigate with a keyboard, or touch screen device.

These requirements for semantic programmatic access parallel User Agent Accessibility Guidelines: Programmatic operation of user agent user interface and Programmatic notification of changes ([UAAG10]) except that it applies to content, not just to the user agent.

2. Using WAI-ARIA §

This section is non-normative.

Complex web applications become inaccessible when assistive technologies cannot determine the semantics behind portions of a document or when the user is unable to effectively navigate to all parts of it in a usable way (see the WAI-ARIA Primer [WAI-ARIA-PRIMER]). WAI-ARIA divides the semantics into roles (the type defining a user interface element) and state and properties supported by the roles.

Authors need to associate elements in the document to a WAI-ARIA role and the appropriate states and properties (aria-* attributes) during its life-cycle, unless the elements already have the appropriate implicit WAI-ARIA semantics for states and properties. In these instances the equivalent host language states and properties take precedence to avoid a conflict while the role attribute will take precedence over the implicit role of the host language element.

2.1 WAI-ARIA Roles §

A WAI-ARIA role is set on an element using a `role` attribute, similar to the `role` attribute defined in the Role Attribute [ROLE-ATTRIBUTE].

EXAMPLE 1

```
<li role="menuitem">Open file…</li>
```

The roles defined in this specification include a collection of document landmarks and the WAI-ARIA role taxonomy.

The roles in this taxonomy and their expected behaviors are modeled using RDF/OWL [OWL-FEATURES]. Features of the role taxonomy provide the following information for each role:

- an informative description of the role;
- hierarchical information about related roles (e.g., a list);
- context of the role (e.g., a listitem is contained inside a list);
- references to related concepts in other specifications;
- use of OWL to provide a type hierarchy allowing for semantic inheritance (similar to a class hierarchy); and
- supported state and properties for each role (e.g., a checkbox supports being checked via aria-checked).

Attaching a role gives assistive technologies information about how to handle each element.

2.2 WAI-ARIA States and Properties §

WAI-ARIA provides a collection of accessibility state and properties which are used to support platform accessibility APIs on various operating system platforms. Assistive technologies may access this information through an exposed user agent DOM or through a mapping to the platform accessibility API. When combined with roles, the user agent can supply the assistive technologies with user interface information to convey to the user at any time. Changes in states or properties will result in a notification to assistive technologies, which could alert the user that a change has occurred.

In the following example, a list item (html:li) has been used to create a checkable menu item, and JavaScript events will capture mouse and keyboard events to toggle value value of aria-checked. A role is used to make the behavior of this simple widget known to the user agent. Attributes that change with user actions (such as aria-checked) are defined in the states and properties section.

EXAMPLE 2

```
<li role="menuitemcheckbox" aria-checked="true">Sort by Last Modified</li>
```

Some accessibility states, called managed states, are controlled by the user agent. Examples of managed state include keyboard focus and selection. Managed states often have corresponding CSS pseudo-classes (such as :focus and ::selection) to define style changes. In contrast, the states in this specification are typically controlled by the author and are called unmanaged states. Some states are managed by the user agent, such as aria-posinset and aria-setsize, but the author can override them if the DOM is incomplete and would cause the user agent calculation to be incorrect. User agents map both managed and unmanaged states to the platform accessibility APIs.

Most modern user agents support CSS attribute selectors ([CSS2]), and can allow the author to create UI changes based on WAI-ARIA attribute information, reducing the amount of scripts necessary to achieve equivalent functionality. In the following example, a CSS selector is used to determine whether or not the text is bold and an image of a check mark is shown, based on the value of the aria-checked attribute.

EXAMPLE 3

```
[aria-checked="true"] { font-weight: bold; }
[aria-checked="true"]::before { background-image: url(checked.gif); }
```

If CSS is not used to toggle the visual representation of the check mark, the author could include additional markup and scripts to manage an image that represents whether or not the menuitemcheckbox is checked.

EXAMPLE 4

```
<li role="menuitemcheckbox" aria-checked="true">
  <img src="checked.gif" role="presentation" alt="">
  <!-- note: additional scripts required to toggle image source -->
  Sort by Last Modified
</li>
```

2.3 Managing Focus §

An application should always have an element with focus when in use, as applications require users to have a place to provide user input. Authors are advised to not destroy the element with focus or scroll it off-screen unless through user intervention. All interactive objects should be focusable. All parts of composite interactive controls need to be focusable or have a documented alternative method to achieve their function, such as a keyboard shortcut. Authors are advised to maintain an obvious, discoverable way, either through tabbing or other standard navigation techniques, for keyboard users to move the focus to any interactive element. See User Agent Accessibility Guidelines, Guideline 9 ([UAAG10], Guideline 9).

When using standard HTML and basic WAI-ARIA widgets, application developers can simply manipulate the tab order or use a script to create keyboard shortcuts to elements in the document. Use of more complex widgets requires the author to manage focus within them. SVG Tiny provides a similar navigation "ring" mechanism that by default follows document order and which should be implemented using system dependent input facilities (the TAB key on most desktop computers). SVG authors may place elements in the navigation order by manipulating the focusable attribute and they may dynamically specify the navigation order by modifying elements navigation attributes.

WAI-ARIA includes a number of "managing container" widgets, also known as "composite" widgets. When appropriate, the container is responsible for tracking the last descendant which was active (the default is usually the first item in the container). It is essential that a container maintain a usable and consistent strategy when focus leaves a container and is then later refocused. While there may be exceptions, it is recommended that when a previously focused container is refocused, the active descendant be the same element as the active descendant when the container was last focused. Exceptions include cases where the contents of a container widget have changed, and widgets like a menubar where the user expects to always return to the first item when focus leaves the menu bar. For example, if the second item of a tree group was the active descendant when the user tabbed out of the tree group, then the second item of the tree group remains the active descendant when the tree group gets focus again. The user may also activate the container by clicking on one of the descendants within it.

When the container or its active descendant has focus, the user may navigate through the container by pressing additional keys, such as the arrow keys, to change the currently active descendant. Any additional press of the main navigation key (generally the TAB key) will move out of the container to the next widget.

For example, a grid may be used as a spreadsheet with thousands of gridcell elements, all of which may not be present in the document at one time. This requires focus to be managed by the container using the aria-activedescendant attribute on the managing container element, or by the container managing the tabindex of its child elements and setting focus on the appropriate child. For more information, see Providing Keyboard Focus in WAI-ARIA Authoring Practices ([WAI-ARIA-PRACTICES]).

Content authors are required to manage focus on the following container roles:

- combobox
- grid
- listbox
- menu
- menubar
- radiogroup
- tree
- treegrid
- tablist

More information on managing focus can be found in the Using Tabindex to Manage Focus Among Widgets section of the WAI-ARIA Authoring Practices [WAI-ARIA-PRACTICES].

3. 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 words MAY, MUST, MUST NOT, SHOULD, and SHOULD NOT are to be interpreted as described in [RFC2119].

This specification indicates whether a section is normative or informative. Classifying a section as normative or informative applies to the entire section. A statement "This section is normative" or "This section is informative" applies to all sub-sections of that section.

Normative sections provide requirements that authors, user agents, and assistive technologies MUST follow for an implementation to conform to this specification.

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.

3.1 Non-interference with the Host Language §

WAI-ARIA processing by the user agent MUST NOT interfere with the normal operation of the built-in features of the host language.

If a CSS selector includes a WAI-ARIA attribute (e.g., `input[aria-invalid="true"]`), user agents MUST update the visual display of any elements matching (or no longer matching) the selector any time the attribute is added/changed/removed in the DOM. The user agent MAY alter the mapping of the host language features into an accessibility API, but the user agent MUST NOT alter the DOM in order to remap WAI-ARIA markup into host language features.

3.2 All WAI-ARIA in DOM §

A conforming user agent which implements a document object model that does not conform to the W3C DOM specification MUST include the content attribute for role and its WAI-ARIA role values, as well as the WAI-ARIA States and Properties in the DOM as specified by the author, even though processing may affect how the elements are exposed to accessibility APIs. Doing so ensures that each role attribute and all WAI-ARIA states and properties, including their values, are in the document in an unmodified form so other tools, such as assistive technologies, can access them. A conforming W3C DOM meets this criteria.

3.3 Assistive Technology Notifications Communicated to Web Applications §

Assistive technologies, such as voice recognition systems and alternate input devices for users with mobility impairments, require the ability to control a web application in a device-independent way. WAI-ARIA state and properties reflect the current state of rich internet application components. The ability for assistive technologies to notify web application of necessary changes is essential because it allows these alternative input solutions to control an application without being dependent on the standard input device which the user is unable to effectively control directly.

User agents MUST provide a method to notify the web application when a change occurs to states or properties in the system accessibility API. Likewise, web application authors SHOULD update the web application accordingly when notified of a change request from the user agent or assistive technology.

> NOTE
>
> Many state and properties can be changed by assistive technologies through existing accessibility APIs by responding to a default action event. For example, the aria-selected state of a tab in a tabpanel can be changed by triggering the default action on the element.

3.4 Conformance Checkers §

Any application or script verifying document conformance or validity SHOULD include a test for all of the normative author requirements in this specification. If testing for a given requirement, conformance checkers MUST issue an error if an author "MUST" requirement isn't met, and MUST issue a warning if an author "SHOULD" requirement isn't met.

4. Important Terms §

This section is non-normative.

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].

An accessible object in the accessibility tree and its descendants in that tree. It does not include objects which have relationships other than parent-child in that tree. For example, it does not include objects linked via aria-flowto unless those objects are also descendants in the accessibility tree.

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.

The action taken when an event, typically initiated by users through an input device, causes an element to fulfill a defined role. The role may be defined for that element by the host language, or by author-defined variables, or both. The role for any given element may be a generic action, or may be unique to that element. For example, the activation behavior of an HTML or SVG <a> element shall be to cause the user agent to traverse the link specified in the href attribute, with the further optional parameter of specifying the browsing context for the traversal (such as the current window or tab, a named window, or a new window); the activation behavior of an HTML <input> element with the type attribute value submit shall be to send the values of the form elements to an author-defined IRI by the author-defined HTTP method.

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.

Event from/to the host operating system via the accessibility API, notifying of a change of input focus.

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.

Translated to platform-specific accessibility APIs as defined in the WAI-ARIA User Agent Implementation Guide. [WAI-ARIA-IMPLEMENTATION]

Hidden

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

Informative

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

Keyboard Accessible

Accessible to the user using a keyboard or assistive technologies that mimic keyboard input, such as a sip and puff tube. References in this document relate to WCAG 2 Guideline 2.1: "Make all functionality available from a keyboard" [WCAG20].

Landmark

A type of region on a page to which the user may want quick access. Content in such a region is different from that of other regions on the page and relevant to a specific user purpose, such as navigating, searching, perusing the primary content, etc.

Live Region

Live regions are perceivable regions of a web page that are typically updated as a result of an external event when user focus may be elsewhere. These regions are not always updated as result of a user interaction. This practice has become commonplace with the growing use of Ajax. Examples of live regions include a chat log, stock ticker, or a sport scoring section that updates periodically to reflect game statistics. Since these asynchronous areas are expected to update outside the user's area of focus, assistive technologies such as screen readers have either been unaware of their existence or unable to process them for the user. WAI-ARIA has provided a collection of properties that allow the author to identify these live regions and how to process them: aria-live, aria-relevant, aria-atomic, and aria-busy. Pre-defined live region roles are listed in the Choosing Between Special Case Live Regions ([WAI-ARIA-PRACTICES], Section 5.3).

Primary Content Element

An implementing host language's primary content element, such as the `body` element in HTML.

Managed State

Accessibility API state that is controlled by the user agent, such as focus and selection. These are contrasted with "unmanaged states" that are typically controlled by the author. Nevertheless, authors can override some managed states, such as aria-posinset and aria-setsize. Many managed states have corresponding CSS pseudo-classes, such as :focus, and pseudo-elements, such as ::selection, that are also updated by the user agent.

Nemeth Braille

The Nemeth Braille Code for Mathematics is a braille code for encoding mathematical and scientific notation. See Nemeth Braille on Wikipedia.

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.

Ontology

A description of the characteristics of classes and how they relate to each other.

Operable

Usable by users in ways they can control. References in this document relate to WCAG 2 Principle 2: content must be operable [WCAG20]. See Keyboard Accessible.

Owned Element

An 'owned element' is any DOM descendant of the element, any element specified as a child via aria-owns, or any DOM descendant of the owned child.

An 'owning element' is any DOM ancestor of the element, or any element with an aria-owns attribute which references the ID of the element.

Perceivable

Presentable to users in ways they can sense. References in this document relate to WCAG 2 Principle 1: content must be perceivable [WCAG20].

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.

Relationship

A connection between two distinct things. Relationships may be of various types to indicate which object labels another, controls another, etc.

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.

The primary element containing non-metadata content. In many languages, this is the document element but in HTML, it is the <body> or <frameset>.

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.

Any document created from a <frame>, <iframe> or similar mechanism. A sub-document may contain a document, an application or any widget such as a calendar pulled in from another server. In the accessible tree there are two accessible objects for this situation—one represents the <frame>/<iframe> element in the parent document, which parents a single accessible object child representing the spawned document contents.

An element specified in a WAI-ARIA relation. For example, in , where "elem1" is the ID for the target element.

Taxonomy

A hierarchical definition of how the characteristics of various classes relate to each other, in which classes inherit the properties of superclasses in the hierarchy. A taxonomy can comprise part of the formal definition of an ontology.

Text node

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

Understandable

Presentable to users in ways they can construct an appropriate meaning. References in this document relate to WCAG 2 Principle 3: Information and the operation of user interface must be understandable [WCAG20].

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.

A reference to a target element in the same document that has a matching ID

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).

5. The Roles Model §

This section defines the WAI-ARIA role taxonomy and describes the characteristics and properties of all roles. A formal RDF/OWL representation of all the information presented here is available in Schemata Appendix.

The roles, their characteristics, the states and properties they support, and specification of how they may be used in markup, shall be considered normative. The RDF/OWL representation used to model the taxonomy shall be considered informative. The RDF/OWL taxonomy may be used as a vehicle to extend WAI-ARIA in the future or by tool manufacturers to validate states and properties applicable to roles per this specification.

Roles are element types and authors MUST NOT change role values over time or with user actions. Authors wishing to change a role MUST do so by deleting the associated element and its children and replacing it with a new element with the appropriate role. Typically, platform accessibility APIs do not provide a vehicle to notify assistive technologies of a role value change, and consequently, assistive technologies may not update their cache with the new role attribute value.

In order to reflect the content in the DOM, user agents SHOULD map the role attribute to the appropriate value in the implemented accessibility API, and user agents SHOULD update the mapping when the role attribute changes.

5.1 Relationships Between Concepts §

The role taxonomy uses the following relationships to relate WAI-ARIA roles to each other and to concepts from other specifications, such as HTML and XForms.

5.1.1 Superclass Role §

Inheritance is expressed in RDF using the RDF Schema 1.1 subClassOf ([RDF-SCHEMA]) property.

RDF Property
rdfs:subClassOf

The role that the current subclassed role extends in the taxonomy. This extension causes all the properties and constraints of the superclass role to propagate to the subclass role. Other than well known stable specifications, inheritance may be restricted to items defined inside this specification, so that external items cannot be changed and affect inherited classes.

5.1.2 Subclass Roles §

RDF Property
<none>

Informative list of roles for which this role is the superclass. This is provided to facilitate reading of the specification but adds no new information.

5.1.3 Related Concepts §

RDF Property
role:relatedConcept

Informative data about a similar or related idea from other specifications. Concepts that are related are not necessarily identical. Related concepts do not inherit properties from each other. Hence if the definition of one concept changes, the properties, behavior, and definition of its related concept is not affected.

For example, a progress bar is like a status indicator. Therefore, the status. However, if the definition of progressbar is not affected.

5.1.4 Base Concept §

RDF Property
role:baseConcept

Informative data about objects that are considered prototypes for the role. Base concept is similar to type, but without inheritance of limitations and properties. Base concepts are designed as a substitute for inheritance for external concepts. A base concept is like a related concept except that the base concept is almost identical to the role definition.

For example, the checkbox defined in this document has similar functionality and anticipated behavior to a checkbox defined in HTML. Therefore, a checkbox has an HTML checkbox as a baseConcept. However, if the original HTML checkbox baseConcept definition is modified, the definition of a checkbox in this document will not be affected, because there is no actual inheritance of the respective type.

5.2 Characteristics of Roles §

Roles are defined and described by their characteristics. Characteristics define the structural function of a role, such as what a role is, concepts behind it, and what instances the role can or must contain. In the case of widgets this also includes how it interacts with the user agent based on mapping to HTML forms and XForms. States and properties from WAI-ARIA that are supported by the role are also indicated.

The roles taxonomy defines the following characteristics. These characteristics are implemented in RDF as properties of the OWL classes that describe the roles.

5.2.1 Abstract Roles §

RDF Property
N/A
Values

Boolean

Abstract roles are the foundation upon which all other WAI-ARIA roles are built. Content authors MUST NOT use abstract roles because they are not implemented in the API binding. User agents MUST NOT map abstract roles to the standard role mechanism of the accessibility API. Abstract roles are provided to help with the following:

1. Organize the role taxonomy and provide roles with a meaning in the context of known concepts.
2. Streamline the addition of roles that include necessary features.

5.2.2 Required States and Properties §

RDF Property
role:requiredState
Values
Any valid RDF object reference, such as a URI.

States and properties specifically required for the role and subclass roles. Content authors MUST provide a non-empty value for required states and properties. Content authors MUST NOT use the value "undefined" for required states and properties.

When an object inherits from multiple ancestors and one ancestor indicates that property is supported while another ancestor indicates that it is required, the property is required in the inheriting object.

> NOTE
>
> A host language attribute with the appropriate implicit WAI-ARIA semantic fulfills this requirement.

5.2.3 Supported States and Properties §

RDF Property
role:supportedState
Values
Any valid RDF object reference, such as a URI.

States and properties specifically applicable to the role and child roles. User agents MUST map all supported states and properties for the role to an accessibility API. Content authors MAY provide values for supported states and properties, but need not in some cases where default values are sufficient.

> NOTE
>
> A host language attribute with the appropriate implicit WAI-ARIA semantic fulfills this requirement.

5.2.4 Inherited States and Properties §

Informative list of properties that are inherited onto a role from superclass roles. States and properties are inherited from superclass roles in the role taxonomy, not from ancestor elements in the DOM tree. These properties are not explicitly defined on the role, as the inheritance of properties is automatic. This information is provided to facilitate reading of the specification. The set of supported states and properties combined with inherited states and properties forms the full set of states and properties supported by the role.

5.2.5 Required Owned Elements §

RDF Property
role:mustContain
Values
Any valid RDF object reference, such as a URI.

Any element that will be owned by the element with this role. For example, an element with the role listitem.

When multiple roles are specified as required owned elements for a role, at least one instance of one required owned element is expected. This specification does not require an instance of each of the listed owned roles. For example, a menu should have at least one instance of a menuitem, menuitemcheckbox, or menuitemradio. The menu role does not require one instance of each.

There may be times that required owned elements are missing, for example, while editing or while loading a data set. When a widget is missing required owned elements due to script execution or loading, authors MUST mark a containing element with aria-busy equal to true. For example, until a page is fully initialized and complete, an author could mark the document element as busy.

> NOTE
>
> A role that has 'required owned elements' does not imply the reverse relationship. While processing of a role may be incomplete without elements of given roles present as descendants, elements with roles in this list do not always have to be found within elements of the given role. See required context role for requirements about the context where elements of a given role will be contained.

> NOTE

An element with a subclass role of the 'required owned element' does not fulfill this requirement. For example, the list role requires ownership of an element using either the listitem or group role. Although the group role is the superclass of row, adding a owned element with a role of row will not fulfill the requirement that list must own a listitem or a group.

NOTE

An element with the appropriate implicit WAI-ARIA semantic fulfills this requirement.

5.2.6 Required Context Role §

RDF Property
role:scope
Values
Any valid RDF object reference, such as a URI.

The required context role defines the owning container where this role is allowed. If a role has a required context, authors MUST ensure that an element with the role is contained inside (or owned by) an element with the required context role. For example, an element with role listitem is only meaningful when contained inside (or owned by) an element with role list.

NOTE

A role that has 'required context role' does not imply the reverse relationship. While an element with the given role needs to appear within an element of the listed role(s) in order to be meaningful, elements of the listed roles do not always need descendant elements of the given role in order to be meaningful. See required owned elements for requirements about elements that require presence of a given descendant to be processed properly.

NOTE

An element with the appropriate implicit WAI-ARIA semantic fulfills this requirement.

5.2.7 Accessible Name Calculation §

5.2.7.1 Name Computation §

Name Computation is defined in the Accessible Name and Description specification [ACCNAME-AAM].

5.2.7.2 Description Computation §

Description Computation is defined in the Accessible Name and Description specification [ACCNAME-AAM].

5.2.7.3 Text Alternative Computation §

Text Alternative Computation is defined in the Accessible Name and Description specification [ACCNAME-AAM].

5.2.8 Presentational Children §

RDF Property
role:childrenArePresentational
Values

Boolean (true | false)

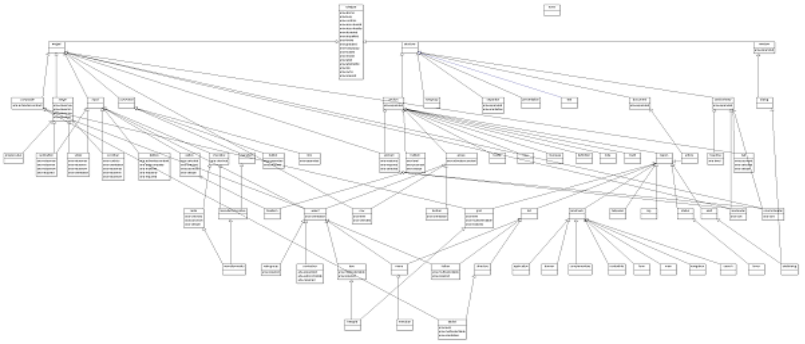
The DOM descendants are presentational. User agents SHOULD NOT expose descendants of this element through the platform accessibility API. If user agents do not hide the descendant nodes, some information may be read twice.

5.2.9 Implicit Value for Role §

Many states and properties have default values. Occasionally, the default value when used on a given role should be different from the usual default. Roles that require a state or property to have a non-standard default value indicate this in the "Implicit Value for Role". This is expressed in the form "state or property name is new default value". Roles that define this have the new default value for the state or property if the author does not provide an explicit value.

5.3 Categorization of Roles §

To support the current user scenario, this specification categorizes roles that define user interface widgets (sliders, tree controls, etc.) and those that define page structure (sections, navigation, etc.). Note that some assistive technologies provide special modes of interaction for regions marked with role application or document.



Class diagram of the relationships described in the role data model.

SVG class diagram | PNG class diagram | Class diagram description

Roles are categorized as follows:

1. Abstract Roles
2. Widget Roles
3. Document Structure Roles
4. Landmark Roles
5. Live Region Roles

5.3.1 Abstract Roles §

The following roles are used to support the WAI-ARIA role taxonomy for the purpose of defining general role concepts.

Abstract roles are used for the ontology. Authors MUST NOT use abstract roles in content.

- command
- composite
- input
- landmark
- range
- roletype
- section
- sectionhead
- select
- structure
- widget
- window

5.3.2 Widget Roles §

The following roles act as standalone user interface widgets or as part of larger, composite widgets.

- alert
- alertdialog
- button
- checkbox
- dialog
- gridcell
- link
- log
- marquee
- menuitem
- menuitemcheckbox
- menuitemradio
- option
- progressbar
- radio
- scrollbar
- searchbox
- slider
- spinbutton
- status
- switch
- tab
- tabpanel
- textbox
- timer
- tooltip
- treeitem

The following roles act as composite user interface widgets. These roles typically act as containers that manage other, contained widgets.

- combobox
- grid
- listbox

- menu
- menubar
- radiogroup
- tablist
- tree
- treegrid

5.3.3 Document Structure §

The following roles describe structures that organize content in a page. Document structures are not usually interactive.

- article
- columnheader
- definition
- directory
- document
- group
- heading
- img
- list
- listitem
- math
- none
- note
- presentation
- region
- row
- rowgroup
- rowheader
- separator
- toolbar

5.3.4 Landmark Roles §

The following roles are regions of the page intended as navigational landmarks. All of these roles inherit from the landmark base type and, with the exception of application, all are imported from the Role Attribute [ROLE-ATTRIBUTE]. The roles are included here in order to make them clearly part of the WAI-ARIA Role taxonomy.

- application
- banner
- complementary
- contentinfo
- form
- main
- navigation
- region
- search

5.3.5 Live Region Roles §

The following widget roles are also live regions and may be modified by live region attributes.

- alert
- log
- marquee
- status
- timer

5.4 Definition of Roles §

Below is an alphabetical list of WAI-ARIA roles to be used by rich internet application authors.

Abstract roles are used for the ontology. Authors MUST NOT use abstract roles in content.

alert
: A type of live region with important, and usually time-sensitive, information. See related alertdialog and status.

alertdialog
: A type of dialog that contains an alert message, where initial focus goes to an element within the dialog. See related alert and dialog.

application
: A region declared as a web application, as opposed to a web document.

article
: A section of a page that consists of a composition that forms an independent part of a document, page, or site.

banner
: A region that contains mostly site-oriented content, rather than page-specific content.

button
: An input that allows for user-triggered actions when clicked or pressed. See related link.

checkbox
: A checkable input that has three possible values: true, false, or mixed.

columnheader
: A cell containing header information for a column.

combobox

A presentation of a select; usually similar to a textbox where users can type ahead to select an option, or type to enter arbitrary text as a new item in the list. See related listbox.

command
A form of widget that performs an action but does not receive input data.

complementary
A supporting section of the document, designed to be complementary to the main content at a similar level in the DOM hierarchy, but remains meaningful when separated from the main content.

composite
A widget that may contain navigable descendants or owned children.

contentinfo
A large perceivable region that contains information about the parent document.

definition
A definition of a term or concept.

dialog
A dialog is a descendant window of the primary window of a web application. For HTML pages, the primary application window is the entire web document, i.e., the body element.

directory
A list of references to members of a group, such as a static table of contents.

document
A region containing related information that is declared as document content, as opposed to a web application.

form
A landmark region that contains a collection of items and objects that, as a whole, combine to create a form. See related search.

grid
An interactive control which contains cells of tabular data arranged in rows and columns, like a table.

gridcell
A cell in a grid or treegrid.

group
A set of user interface objects which are not intended to be included in a page summary or table of contents by assistive technologies.

heading
A heading for a section of the page.

img
A container for a collection of elements that form an image.

input
A generic type of widget that allows user input.

landmark
A perceivable section containing content that is relevant to a specific, author-specified purpose and sufficiently important that users will likely want to be able to navigate to the section easily and to have it listed in a summary of the page. Such a page summary could be generated dynamically by a user agent or assistive technology.

link
An interactive reference to an internal or external resource that, when activated, causes the user agent to navigate to that resource. See related button.

list
A group of non-interactive list items. See related listbox.

listbox
A widget that allows the user to select one or more items from a list of choices. See related combobox and list.

listitem
A single item in a list or directory.

log
A type of live region where new information is added in meaningful order and old information may disappear. See related marquee.

main
The main content of a document.

marquee
A type of live region where non-essential information changes frequently. See related log.

math
Content that represents a mathematical expression.

menu
A type of widget that offers a list of choices to the user.

menubar
A presentation of menu that usually remains visible and is usually presented horizontally.

menuitem
An option in a set of choices contained by a menu or menubar.

menuitemcheckbox
A menuitem with a checkable state whose possible values are true, false, or mixed.

menuitemradio
A checkable menuitem in a set of elements with the same role, only one of which can be checked at a time.

navigation
A collection of navigational elements (usually links) for navigating the document or related documents.

none
[ARIA 1.1] An element whose implicit native role semantics will not be mapped to the accessibility API. See synonym presentation [ARIA 1.0].

note
A section whose content is parenthetic or ancillary to the main content of the resource.

option
A selectable item in a select list.

presentation
An element whose implicit native role semantics will not be mapped to the accessibility API. See synonym none [ARIA 1.1].

progressbar
An element that displays the progress status for tasks that take a long time.

radio
: A checkable input in a group of elements with the same role, only one of which can be checked at a time.

radiogroup
: A group of radio buttons.

range
: An input representing a range of values that can be set by the user.

region
: A perceivable section containing content that is relevant to a specific, author-specified purpose and sufficiently important that users will likely want to be able to navigate to the section easily and to have it listed in a summary of the page. Such a page summary could be generated dynamically by a user agent or assistive technology.

roletype
: The base role from which all other roles in this taxonomy inherit.

row
: A row of cells in a grid.

rowgroup
: A structure containing one or more row elements in a grid.

rowheader
: A cell containing header information for a row in a grid.

search
: A landmark region that contains a collection of items and objects that, as a whole, combine to create a search facility. See related form and searchbox.

searchbox
: [ARIA 1.1] A type of textbox intended for specifying search criteria. See related textbox and search.

section
: A renderable structural containment unit in a document or application.

sectionhead
: A structure that labels or summarizes the topic of its related section.

select
: A form widget that allows the user to make selections from a set of choices.

separator
: A divider that separates and distinguishes sections of content or groups of menuitems.

scrollbar
: A graphical object that controls the scrolling of content within a viewing area, regardless of whether the content is fully displayed within the viewing area.

slider
: A user input where the user selects a value from within a given range.

spinbutton
: A form of range that expects the user to select from among discrete choices.

status
: A type of live region whose content is advisory information for the user but is not important enough to justify an alert, often but not necessarily presented as a status bar. See related alert.

structure
: A document structural element.

switch
: [ARIA 1.1] A type of checkbox that represents on/off values, as opposed to checked/unchecked values. See related checkbox.

tab
: A grouping label providing a mechanism for selecting the tab content that is to be rendered to the user.

tablist
: A list of tab elements, which are references to tabpanel elements.

tabpanel
: A container for the resources associated with a tab, where each tab is contained in a tablist.

text
: [ARIA 1.1] An element whose entire subtree should be exposed to accessibility APIs as plain text.

textbox
: A type of input that allows free-form text as its value.

timer
: A type of live region containing a numerical counter which indicates an amount of elapsed time from a start point, or the time remaining until an end point.

toolbar
: A collection of commonly used function buttons or controls represented in compact visual form.

tooltip
: A contextual popup that displays a description for an element.

tree
: A type of list that may contain sub-level nested groups that can be collapsed and expanded.

treegrid
: A grid whose rows can be expanded and collapsed in the same manner as for a tree.

treeitem
: An option item of a tree. This is an element within a tree that may be expanded or collapsed if it contains a sub-level group of tree item elements.

widget
: An interactive component of a graphical user interface (GUI).

window
: A browser or application window.

---

alert (role) §

A type of live region with important, and usually time-sensitive, information. See related alertdialog and status.

Alerts are used to convey messages to alert the user. In the case of audio warnings this is an accessible alternative for a hearing-impaired user. The alert role goes on the node containing the alert message. Alerts are specialized forms of the

status role, which will be processed as an atomic live region.

Alerts are assertive live regions and will be processed as such by assistive technologies. Neither authors nor user agents are required to set or manage focus to them in order for them to be processed. Since alerts are not required to receive focus, content authors SHOULD NOT require users to close an alert. If the operating system allows, the user agent SHOULD fire a system alert event through the accessibility API when the WAI-ARIA alert is created. If an alert requires focus to close the alert, then content authors SHOULD use alertdialog instead.

> NOTE
>
> Elements with the role alert have an implicit aria-live value of assertive, and an implicit aria-atomic value of true.

Characteristics:

Characteristic	Value
Superclass Role:	section
Subclass Roles:	alertdialog
Related Concepts:	XForms alert
Inherited States and Properties:	aria-atomic aria-busy (state) aria-controls aria-current (state) aria-describedat aria-describedby aria-disabled (state) aria-dropeffect aria-expanded (state) aria-flowto aria-grabbed (state) aria-haspopup aria-hidden (state) aria-invalid (state) aria-label aria-labelledby aria-live aria-owns aria-relevant
Name From:	author
Implicit Value for Role:	Default for aria-live is assertive. Default for aria-atomic is true.

---

alertdialog (role) §

A type of dialog that contains an alert message, where initial focus goes to an element within the dialog. See related alert and dialog.

Alert dialogs are used to convey messages to alert the user. The alertdialog role goes on the node containing both the alert message and the rest of the dialog. Content authors SHOULD make alert dialogs modal by ensuring that, while the alertdialog is shown, keyboard and mouse interactions only operate within the dialog. See aria-modal [ARIA 1.1].

Unlike alert, alertdialog can receive a response from the user. For example, to confirm that the user understands the alert being generated. When the alert dialog is displayed, authors SHOULD set focus to an active element within the alert dialog, such as a form edit field or an OK button. The user agent SHOULD fire a system alert event through the accessibility API when the alert is created, provided one is specified by the intended accessibility API.

Authors SHOULD use aria-describedby on an alertdialog to reference the alert message element in the dialog. If they do not, an assistive technology can resort to its internal recovery mechanism to determine the contents of the alert message.

Characteristics:

Characteristic	Value
Superclass Role:	alert dialog
Related Concepts:	XForms alert
Inherited States and Properties:	aria-atomic aria-busy (state) aria-controls aria-current (state) aria-describedat aria-describedby aria-disabled (state) aria-dropeffect aria-expanded (state) aria-flowto aria-grabbed (state) aria-haspopup aria-hidden (state) aria-invalid (state)

	aria-label aria-labelledby aria-live aria-modal aria-owns aria-relevant
Name From:	author
Accessible Name Required:	True

application (role) §

A region declared as a web application, as opposed to a web document.

When the user navigates an element assigned the role of application, assistive technologies that typically intercept standard keyboard events SHOULD switch to an application browsing mode, and pass keyboard events through to the web application. The intent is to hint to certain assistive technologies to switch from normal browsing mode into a mode more appropriate for interacting with a web application; some user agents have a browse navigation mode where keys, such as up and down arrows, are used to browse the document, and this native behavior prevents the use of these keys by a web application.

> NOTE
>
> Where appropriate, assistive technologies that typically intercept other standard device input events, such as touch screen input, could switch to an application browsing mode that passes some or all of those events through to the web application.

Authors SHOULD set the role of application on the element that encompasses the entire application. If the application role applies to the entire web page, authors SHOULD set the role of application on the root node for content, such as the body element in HTML or svg element in SVG.

For example, an email application has a document and an application in it. The author would want to use typical application navigation mode to cycle through the list of emails, and much of this navigation would be defined by the application author. However, when reading an email message the content will appear in a region with a document role in order to use browsing navigation.

For all instances of non-decorative static text or image content inside an application, authors SHOULD either associate the text with a form widget or group (via aria-label, aria-labelledby, or aria-describedby) or separate the text into an element with role of document or article.

Authors SHOULD provide a title or label for applications. Authors SHOULD use label text that is suitable for use as a navigation preview or table-of-contents entry for the page section. Content authors SHOULD provide the label through one of the following methods:

- If the application includes the entire contents of the web page, use the host language feature for title or label, such as the title element in both HTML and SVG. This has the effect of labeling the entire application.
- Otherwise, provide a visible label referenced by the application using aria-labelledby.

User agents SHOULD treat elements with the role of application as navigational landmarks.

Authors MAY use the application role on the primary content element of the host language (such as the body element in HTML) to define an entire page as an application. However, if the primary content element is defined as having a role of application, user agents MUST NOT use the element as a navigational landmark. If assistive technologies use an interaction mode that intercepts standard keyboard events, when encountering the application role, those assistive technologies SHOULD switch to an interaction mode that passes keyboard events through to the web application.

Characteristics:

Characteristic	Value
Superclass Role:	landmark
Related Concepts:	Device Independence Delivery Unit
Inherited States and Properties:	aria-atomic aria-busy (state) aria-controls aria-current (state) aria-describedat aria-describedby aria-disabled (state) aria-dropeffect aria-expanded (state) aria-flowto aria-grabbed (state) aria-haspopup aria-hidden (state) aria-invalid (state) aria-label aria-labelledby aria-live aria-owns aria-relevant
Name From:	author

Accessible Name Required:	True

article (role) §

A section of a page that consists of a composition that forms an independent part of a document, page, or site.

An article is not a navigational landmark, but may be nested to form a discussion where assistive technologies could pay attention to article nesting to assist the user in following the discussion. An article could be a forum post, a magazine or newspaper article, a web log entry, a user-submitted comment, or any other independent item of content. It is independent in that its contents could stand alone, for example in syndication. However, the element is still associated with its ancestors; for instance, contact information that applies to a parent body element still covers the article as well. When nesting articles, the child articles represent content that is related to the content of the parent article. For instance, a web log entry on a site that accepts user-submitted comments could represent the comments as articles nested within the article for the web log entry. Author, heading, date, or other information associated with an article does not apply to nested articles.

When the user navigates an element assigned the role of article, assistive technologies that typically intercept standard keyboard events SHOULD switch to document browsing mode, as opposed to passing keyboard events through to the web application. Assistive technologies MAY provide a feature allowing the user to navigate the hierarchy of any nested article elements.

Characteristics:

Characteristic	Value
Superclass Role:	document
Related Concepts:	HTML 5 article
Inherited States and Properties:	aria-atomic aria-busy (state) aria-controls aria-current (state) aria-describedat aria-describedby aria-disabled (state) aria-dropeffect aria-expanded (state) aria-flowto aria-grabbed (state) aria-haspopup aria-hidden (state) aria-invalid (state) aria-label aria-labelledby aria-live aria-owns aria-relevant
Name From:	author

banner (role) §

A region that contains mostly site-oriented content, rather than page-specific content.

Site-oriented content typically includes things such as the logo or identity of the site sponsor, and site-specific search tool. A banner usually appears at the top of the page and typically spans the full width.

User agents SHOULD treat elements with the role of banner as navigational landmarks.

Within any document or application, the author SHOULD mark no more than one element with the banner role.

> NOTE
>
> Because document and application elements can be nested in the DOM, they may have multiple banner elements as DOM descendants, assuming each of those is associated with different document nodes, either by a DOM nesting (e.g., document within document) or by use of the aria-owns attribute.

Characteristics:

Characteristic	Value
Superclass Role:	landmark
Inherited States and Properties:	aria-atomic aria-busy (state) aria-controls aria-current (state) aria-describedat aria-describedby aria-disabled (state) aria-dropeffect aria-expanded (state)

	aria-flowto aria-grabbed (state) aria-haspopup aria-hidden (state) aria-invalid (state) aria-label aria-labelledby aria-live aria-owns aria-relevant
Name From:	author

button (role) §

An input that allows for user-triggered actions when clicked or pressed. See related link.

Buttons are mostly used for discrete actions. Standardizing the appearance of buttons enhances the user's recognition of the widgets as buttons and allows for a more compact display in toolbars.

Buttons support the optional attribute aria-pressed. Buttons with a non-empty aria-pressed attribute are toggle buttons. When aria-pressed is true the button is in a "pressed" state, when aria-pressed is false it is not pressed. If the attribute is not present, the button is a simple command button.

Characteristics:

Characteristic	Value
Superclass Role:	command
Base Concept:	HTML button
Related Concepts:	link XForms trigger
Supported States and Properties:	aria-expanded aria-pressed
Inherited States and Properties:	aria-atomic aria-busy (state) aria-controls aria-current (state) aria-describedat aria-describedby aria-disabled (state) aria-dropeffect aria-flowto aria-grabbed (state) aria-haspopup aria-hidden (state) aria-invalid (state) aria-label aria-labelledby aria-live aria-owns aria-relevant
Name From:	contents author
Accessible Name Required:	True
Children Presentational:	True

checkbox (role) §

A checkable input that has three possible values: true, false, or mixed.

The aria-checked attribute of a checkbox indicates whether the input is checked (true), unchecked (false), or represents a group of elements that have a mixture of checked and unchecked values (mixed). Many checkboxes do not use the mixed value, and thus are effectively boolean checkboxes.

Characteristics:

Characteristic	Value
Superclass Role:	input
Subclass Roles:	menuitemcheckbox radio switch
Related Concepts:	HTML input[type="checkbox"] option
Required States and Properties:	aria-checked
Inherited States and Properties:	aria-atomic aria-busy (state) aria-controls aria-current (state)

	aria-describedat aria-describedby aria-disabled (state) aria-dropeffect aria-flowto aria-grabbed (state) aria-haspopup aria-hidden (state) aria-invalid (state) aria-label aria-labelledby aria-live aria-owns aria-relevant
Name From:	contents author
Accessible Name Required:	True
Implicit Value for Role:	Default for aria-checked is false.

columnheader (role) §

A cell containing header information for a column.

columnheader can be used as a column header in a table or grid. It could also be used in a pie chart to show a similar relationship in the data.

The columnheader establishes a relationship between it and all cells in the corresponding column. It is the structural equivalent to an HTML th element with a column scope.

Authors MUST ensure elements with role columnheader are contained in, or owned by, an element with the role row.

Applying the aria-selected state on a columnheader MUST not cause the user agent to automatically propogate the aria-selected state to all the cells in the corresponding column. An author MAY choose to propogate selection in this manner depending on the specific application.

> NOTE
>
> Because cells are organized into rows, there is not a single container element for the column. The column is the set of gridcell elements in a particular position within their respective row containers.

Characteristics:

Characteristic	Value
Superclass Role:	gridcell sectionhead widget
Base Concept:	HTML th[scope="col"]
Required Context Role:	row
Supported States and Properties:	aria-sort
Inherited States and Properties:	aria-atomic aria-busy (state) aria-colindex aria-colspan aria-controls aria-current (state) aria-describedat aria-describedby aria-disabled (state) aria-dropeffect aria-expanded (state) aria-flowto aria-grabbed (state) aria-haspopup aria-hidden (state) aria-invalid (state) aria-label aria-labelledby aria-live aria-owns aria-readonly aria-relevant aria-required aria-rowindex aria-rowspan aria-selected (state)
Name From:	contents author

Accessible Name Required:	True

combobox (role) §

A presentation of a select; usually similar to a textbox where users can type ahead to select an option, or type to enter arbitrary text as a new item in the list. See related listbox.

combobox is the combined presentation of a single line textfield with a listbox popup. The combobox may be editable. Typically editable combo boxes are used for autocomplete behavior, and authors SHOULD set aria-autocomplete attribute on the textfield.

- If an author sets a combobox's value of aria-autocomplete to 'none' (default), authors MUST manage and set focus on the associated listbox, so assistive technologies can follow the currently selected value.
- If an author sets a combobox's value of aria-autocomplete to 'inline' or 'both', authors MUST update the value of the focused textfield, so assistive technologies can announce the currently selected value.
- If an author sets a combobox's value of aria-autocomplete to 'list', user agents MUST expose changes to the aria-activedescendant attribute on the combobox while the combobox remains focused. If a change to the aria-activedescendant attribute occurs while the combobox is focused, assistive technologies SHOULD alert the user of that change, for example, by speaking the text alternative of the new active descendant element. Authors SHOULD associate the combobox textfield with its listbox using aria-owns. For example:

EXAMPLE 5

```
<input type="text" aria-label="Tag" role="combobox" aria-expanded="true"
  aria-autocomplete="list" aria-owns="owned_listbox" aria-activedescendant="selected_option">
<ul role="listbox" id="owned_listbox">
  <li role="option">Zebra</li>
  <li role="option" id="selected_option">Zoom</li>
</ul>
```

NOTE

In XForms [XFORMS10] the same select can have one of 3 appearances: combo-box, drop-down box, or group of radio-buttons. Many browsers allow users to type ahead to existing choices in a drop-down select widget. This specification does not constrain the presentation of the combo box.

To be keyboard accessible, authors SHOULD manage focus of descendants for all instances of this role, as described in Managing Focus.

NOTE

Elements with the role combobox have an implicit aria-expanded value of false.

NOTE

Elements with the role combobox have an implicit aria-haspopup value of true.

NOTE

Elements with the role combobox have an implicit aria-orientation value of vertical.

Characteristics:

Characteristic	Value
Superclass Role:	select
Related Concepts:	HTML select XForms select
Required Owned Elements:	listbox textbox
Required States and Properties:	aria-expanded
Supported States and Properties:	aria-autocomplete aria-required
Inherited States and Properties:	aria-activedescendant aria-atomic aria-busy (state) aria-controls aria-current (state) aria-describedat aria-describedby aria-disabled (state) aria-dropeffect aria-expanded (state)

	aria-flowto aria-grabbed (state) aria-haspopup aria-hidden (state) aria-invalid (state) aria-label aria-labelledby aria-live aria-orientation aria-owns aria-relevant
Name From:	author
Accessible Name Required:	True
Implicit Value for Role:	Default for aria-expanded is false. Default for aria-haspopup is true. Default for aria-orientation is vertical.

command (abstract role) §

A form of widget that performs an action but does not receive input data.

> NOTE
>
> command is an abstract role used for the ontology. Authors should not use this role in content.

Characteristics:

Characteristic	Value
Is Abstract:	True
Superclass Role:	widget
Subclass Roles:	button link menuitem
Related Concepts:	HTML 5 command
Inherited States and Properties:	aria-atomic aria-busy (state) aria-controls aria-current (state) aria-describedat aria-describedby aria-disabled (state) aria-dropeffect aria-flowto aria-grabbed (state) aria-haspopup aria-hidden (state) aria-invalid (state) aria-label aria-labelledby aria-live aria-owns aria-relevant
Name From:	author

complementary (role) §

A supporting section of the document, designed to be complementary to the main content at a similar level in the DOM hierarchy, but remains meaningful when separated from the main content.

There are various types of content that would appropriately have this role. For example, in the case of a portal, this may include but not be limited to show times, current weather, related articles, or stocks to watch. The complementary role indicates that contained content is relevant to the main content. If the complementary content is completely separable main content, it may be appropriate to use a more general role.

User agents SHOULD treat elements with the role of complementary as navigational landmarks.

Characteristics:

Characteristic	Value
Superclass Role:	landmark
Inherited States and Properties:	aria-atomic aria-busy (state) aria-controls aria-current (state)

	aria-describedat aria-describedby aria-disabled (state) aria-dropeffect aria-expanded (state) aria-flowto aria-grabbed (state) aria-haspopup aria-hidden (state) aria-invalid (state) aria-label aria-labelledby aria-live aria-owns aria-relevant
Name From:	author

composite (abstract role) §

A widget that may contain navigable descendants or owned children.

Authors SHOULD ensure that a composite widget exist as a single navigation stop within the larger navigation system of the web page. Once the composite widget has focus, authors SHOULD provide a separate navigation mechanism for users to navigate to elements that are descendants or owned children of the composite element.

> NOTE
>
> composite is an abstract role used for the ontology. Authors should not use this role in content.

Characteristics:

Characteristic	Value
Is Abstract:	True
Superclass Role:	widget
Subclass Roles:	grid select tablist
Supported States and Properties:	aria-activedescendant
Inherited States and Properties:	aria-atomic aria-busy (state) aria-controls aria-current (state) aria-describedat aria-describedby aria-disabled (state) aria-dropeffect aria-flowto aria-grabbed (state) aria-haspopup aria-hidden (state) aria-invalid (state) aria-label aria-labelledby aria-live aria-owns aria-relevant
Name From:	author

contentinfo (role) §

A large perceivable region that contains information about the parent document.

Examples of information included in this region of the page are copyrights and links to privacy statements.

User agents SHOULD treat elements with the role of contentinfo as navigational landmarks.

Within any document or application, the author SHOULD mark no more than one element with the contentinfo role.

> NOTE
>
> Because document and application elements can be nested in the DOM, they may have multiple contentinfo elements as DOM descendants, assuming each of those is associated with different document nodes, either by a DOM nesting (e.g., aria-owns attribute.

Characteristics:

Characteristic	Value
Superclass Role:	landmark
Inherited States and Properties:	aria-atomic aria-busy (state) aria-controls aria-current (state) aria-describedat aria-describedby aria-disabled (state) aria-dropeffect aria-expanded (state) aria-flowto aria-grabbed (state) aria-haspopup aria-hidden (state) aria-invalid (state) aria-label aria-labelledby aria-live aria-owns aria-relevant
Name From:	author

definition (role) §

A definition of a term or concept.

The WAI-ARIA specification does not provide a role to specify the definition term, but host languages may provide such an element. If a host language has an appropriate element for the term (e.g., dfn or dt in HTML), authors SHOULD include the term in that element. Authors SHOULD identify the definition term by using an aria-labelledby attribute on each element with a role of definition.

Characteristics:

Characteristic	Value
Superclass Role:	section
Inherited States and Properties:	aria-atomic aria-busy (state) aria-controls aria-current (state) aria-describedat aria-describedby aria-disabled (state) aria-dropeffect aria-expanded (state) aria-flowto aria-grabbed (state) aria-haspopup aria-hidden (state) aria-invalid (state) aria-label aria-labelledby aria-live aria-owns aria-relevant
Name From:	author

dialog (role) §

A dialog is a descendant window of the primary window of a web application. For HTML pages, the primary application window is the entire web document, i.e., the body element.

Dialogs are most often used to prompt the user to enter or respond to information. A dialog that is designed to interrupt workflow is usually modal. See related alertdialog.

Authors SHOULD provide a dialog label, which can be done with the aria-label or aria-labelledby attribute.

Authors SHOULD ensure that all dialogs (both modal and non-modal) have at least one focusable descendant element. Authors SHOULD focus an element in the modal dialog when it is displayed, and authors SHOULD manage focus of modal dialogs.

> NOTE
>
> In the description of this role, the term "web application" does not refer to the application role, which specifies specific assistive technology behaviors.

Characteristics:

Characteristic	Value
Superclass Role:	window
Subclass Roles:	alertdialog
Inherited States and Properties:	aria-atomic aria-busy (state) aria-controls aria-current (state) aria-describedat aria-describedby aria-disabled (state) aria-dropeffect aria-expanded (state) aria-flowto aria-grabbed (state) aria-haspopup aria-hidden (state) aria-invalid (state) aria-label aria-labelledby aria-live aria-modal aria-owns aria-relevant
Name From:	author
Accessible Name Required:	True

directory (role) §

A list of references to members of a group, such as a static table of contents.

Authors SHOULD use this role for a static table of contents, whether linked or unlinked. This includes tables of contents built with lists, including nested lists. Dynamic tables of contents, however, might use a tree role instead.

Characteristics:

Characteristic	Value
Superclass Role:	list
Subclass Roles:	tablist
Related Concepts:	DAISY Guide
Inherited States and Properties:	aria-atomic aria-busy (state) aria-controls aria-current (state) aria-describedat aria-describedby aria-disabled (state) aria-dropeffect aria-expanded (state) aria-flowto aria-grabbed (state) aria-haspopup aria-hidden (state) aria-invalid (state) aria-label aria-labelledby aria-live aria-owns aria-relevant
Name From:	author

document (role) §

A region containing related information that is declared as document content, as opposed to a web application.

When the user navigates an element assigned the role of document, assistive technologies that typically intercept standard keyboard events SHOULD switch to document browsing mode, as opposed to passing keyboard events through to the web application. The document role informs user agents of the need to augment browser keyboard support in order to allow users to visit and read any content within the document region. In contrast, additional commands are not necessary for screen reader users to read text within a region with the application role, where if coded in an accessible manner, all text will be semantically associated with focusable elements. An important trait of documents is that they have text which is not associated with widgets or groups thereof.

Authors SHOULD set the role of document on the element that encompasses the entire document. If the document role applies to the entire web page, authors SHOULD set the role of document on the root node for content, such as the body element in HTML or svg element in SVG.

For example, an email application has a document and an application in it. The author would want to use typical application navigation mode to cycle through the list of emails, and much of this navigation would be defined by the application

author. However, when reading an email message, the content will appear in a region with a document role in order to use browsing navigation.

Authors SHOULD provide a title or label for documents. Authors SHOULD use label text that suitable for use as a navigation preview or table-of-contents entry for the page section. Content authors SHOULD provide the label through one of the following methods:

- If the document includes the entire contents of the web page, use the host language feature for title or label, such as the title element in both HTML and SVG. This has the effect of labeling the entire document.
- Otherwise, provide a visible label referenced by the document using aria-labelledby.

Characteristics:

Characteristic	Value
Superclass Role:	structure
Subclass Roles:	article
Related Concepts:	Device Independence Delivery Unit
Supported States and Properties:	aria-expanded
Inherited States and Properties:	aria-atomic aria-busy (state) aria-controls aria-current (state) aria-describedat aria-describedby aria-disabled (state) aria-dropeffect aria-flowto aria-grabbed (state) aria-haspopup aria-hidden (state) aria-invalid (state) aria-label aria-labelledby aria-live aria-owns aria-relevant
Name From:	author
Accessible Name Required:	True

---

form (role) §

A landmark region that contains a collection of items and objects that, as a whole, combine to create a form. See related search.

A form may be a mix of host language form controls, scripted controls, and hyperlinks. Authors are reminded to use native host language semantics to create form controls, whenever possible. For search facilities, authors SHOULD use the search role and not the generic form role. Authors SHOULD provide a visible label for the form referenced with aria-labelledby. If an author uses a script to submit a form based on a user action that would otherwise not trigger an onsubmit event (for example, a form submission triggered by the user changing a form element's value), the author SHOULD provide the user with advance notification of the behavior. Authors are reminded to use native host language semantics to create form controls, whenever possible.

User agents SHOULD treat elements with the role of form as navigational landmarks.

Characteristics:

Characteristic	Value
Superclass Role:	landmark
Base Concept:	HTML form
Required States and Properties:	
Inherited States and Properties:	aria-atomic aria-busy (state) aria-controls aria-current (state) aria-describedat aria-describedby aria-disabled (state) aria-dropeffect aria-expanded (state) aria-flowto aria-grabbed (state) aria-haspopup aria-hidden (state) aria-invalid (state) aria-label aria-labelledby aria-live aria-owns aria-relevant

Name From:	author

grid (role) §

An interactive control which contains cells of tabular data arranged in rows and columns, like a table.

> EDITOR'S NOTE
>
> JC 2014-06-03. Now that 1.1 will include a table role, we need to differentiate the short descrition more sufficiently. Emphasize that grids are "interactive" tables that maintain a selection state.

Grids do not necessarily imply presentation. The grid construct describes relationships between data such that it may be used for different presentations. Grids allow the user to move focus between cells using two dimensional navigation.

Authors MUST ensure that elements with role treegrid are owned by elements with role row, which are in turn owned by an element with role rowgroup, grid, or treegrid. If the author applies any non-global WAI-ARIA states or properties to a native markup element that is acting as a row (such as the tr element in HTML), the author MUST also apply the role of row, as stated in the section on Implementation in Host Languages. Authors MAY make cells focusable. Authors MAY provide row and column headers for grids, by using rowheader and columnheader roles.

Since WAI-ARIA can augment an element in the host language, grids can reuse existing functionality of native table grids. When WAI-ARIA grid or gridcell roles overlay host language table elements they reuse the host language semantics for that table. For instance, WAI-ARIA does not specify general attributes for gridcell elements that span multiple rows or columns. When the author needs a gridcell to span multiple rows or columns, use the host language markup, such as the colspan and rowspan attributes in HTML.

Authors MAY determine the contents of a gridcell through calculation of a mathematical formula. Authors MAY make a cell's formula editable by the user. In a spreadsheet application for example, the text alternative of a cell may be the calculated value of a formula. However, when the cell is being edited, the text alternative may be the formula itself.

gridcell elements with the aria-selected attribute set can be selected for user interaction, and if the aria-multiselectable attribute of the grid is set to true, multiple cells in the grid may be selected. Grids may be used for spreadsheets like those in desktop spreadsheet applications.

A grid is considered editable unless otherwise specified. To make a grid read-only, sert thearia-readonly attribute of the grid to true. User Agents MUST implicitly propagate value of the treegrid element's aria-readonly attribute to all of its owned gridcell elements, and expose this to the accessibility API. An author MAY override an individual gridcell element's propagated aria-readonly attribute on the gridcell.

To be keyboard accessible, authors SHOULD manage focus of descendants for all instances of this role, as described in Managing Focus.

Characteristics:

Characteristic	Value
Superclass Role:	composite section
Subclass Roles:	treegrid
Base Concept:	HTML table
Required Owned Elements:	row rowgroup → row
Required States and Properties:	
Supported States and Properties:	aria-colcount aria-level aria-multiselectable aria-readonly aria-rowcount
Inherited States and Properties:	aria-activedescendant aria-atomic aria-busy (state) aria-controls aria-current (state) aria-describedat aria-describedby aria-disabled (state) aria-dropeffect aria-expanded (state) aria-flowto aria-grabbed (state) aria-haspopup aria-hidden (state) aria-invalid (state) aria-label aria-labelledby aria-live aria-owns aria-relevant
Name From:	author

Accessible Name Required:	True

gridcell (role) §

A cell in a grid or treegrid.

Cells may be active, editable, and selectable. Cells may have relationships such as aria-controls to address the application of functional relationships.

If relevant headers cannot be determined from the DOM structure, authors SHOULD explicitly indicate which header cells are relevant to the cell by referencing elements with role rowheader or columnheader using the aria-describedby attribute.

In a treegrid, authors MAY define cells as expandable by using the aria-expanded attribute. If the aria-expanded attribute is provided, it applies only to the individual cell. It is not a proxy for the container row, which also can be expanded. The main use case for providing this attribute on a cell is pivot table behavior.

Authors MUST ensure elements with role gridcell are contained in, or owned by, an element with the role row.

Characteristics:

Characteristic	Value
Superclass Role:	section widget
Subclass Roles:	columnheader rowheader
Base Concept:	HTML td
Required Context Role:	row
Supported States and Properties:	aria-colindex aria-colspan aria-readonly aria-required aria-rowindex aria-rowspan aria-selected
Inherited States and Properties:	aria-atomic aria-busy (state) aria-controls aria-current (state) aria-describedat aria-describedby aria-disabled (state) aria-dropeffect aria-expanded (state) aria-flowto aria-grabbed (state) aria-haspopup aria-hidden (state) aria-invalid (state) aria-label aria-labelledby aria-live aria-owns aria-relevant
Name From:	contents author
Accessible Name Required:	True

group (role) §

A set of user interface objects which are not intended to be included in a page summary or table of contents by assistive technologies.

Contrast with region which is a grouping of user interface objects that will be included in a page summary or table of contents.

Authors SHOULD use a group to form logical collection of items in a widget such as children in a tree widget forming a collection of siblings in a hierarchy, or a collection of items having the same container in a directory. However, when a group is used in the context of list, authors MUST limit its children to listitem elements. Therefore, proper handling of group by authors and assistive technologies is determined by the context in which it is provided.

Authors MAY nest group elements. If a section is significant enough to warrant inclusion in the web page's table of contents, the author SHOULD assign the section a role of region or a standard landmark role.

Characteristics:

Characteristic	Value
Superclass Role:	section
Subclass Roles:	row select

	toolbar
Related Concepts:	HTML fieldset
Supported States and Properties:	aria-activedescendant
Inherited States and Properties:	aria-atomic aria-busy (state) aria-controls aria-current (state) aria-describedat aria-describedby aria-disabled (state) aria-dropeffect aria-expanded (state) aria-flowto aria-grabbed (state) aria-haspopup aria-hidden (state) aria-invalid (state) aria-label aria-labelledby aria-live aria-owns aria-relevant
Name From:	author

heading (role) §

A heading for a section of the page.

Often, heading elements will be referenced with the aria-labelledby attribute of the section for which they serve as a heading. If headings are organized into a logical outline, the aria-level attribute can be used to indicate the nesting level.

Characteristics:

Characteristic	Value
Superclass Role:	sectionhead
Related Concepts:	HTML h1 HTML h2 HTML h3 HTML h4 HTML h5 HTML h6 DTD levelhd
Supported States and Properties:	aria-level
Inherited States and Properties:	aria-atomic aria-busy (state) aria-controls aria-current (state) aria-describedat aria-describedby aria-disabled (state) aria-dropeffect aria-expanded (state) aria-flowto aria-grabbed (state) aria-haspopup aria-hidden (state) aria-invalid (state) aria-label aria-labelledby aria-live aria-owns aria-relevant
Name From:	contents author
Accessible Name Required:	True

img (role) §

A container for a collection of elements that form an image.

An img can contain captions and descriptive text, as well as multiple image files that when viewed together give the impression of a single image. An img represents a single graphic within a document, whether or not it is formed by a collection of drawing objects. In order for elements with a role of img be perceivable, authors MUST provide alternative text or a label determined by the accessible name calculation.

Characteristics:

Characteristic	Value
Superclass Role:	section
Related Concepts:	DTB imggroup HTML img
Inherited States and Properties:	aria-atomic aria-busy (state) aria-controls aria-current (state) aria-describedat aria-describedby aria-disabled (state) aria-dropeffect aria-expanded (state) aria-flowto aria-grabbed (state) aria-haspopup aria-hidden (state) aria-invalid (state) aria-label aria-labelledby aria-live aria-owns aria-relevant
Name From:	author
Accessible Name Required:	True
Children Presentational:	True

---

input (abstract role) §

A generic type of widget that allows user input.

Characteristics:

Characteristic	Value
Is Abstract:	True
Superclass Role:	widget
Subclass Roles:	checkbox option scrollbar select slider spinbutton textbox
Related Concepts:	XForms input
Inherited States and Properties:	aria-atomic aria-busy (state) aria-controls aria-current (state) aria-describedat aria-describedby aria-disabled (state) aria-dropeffect aria-flowto aria-grabbed (state) aria-haspopup aria-hidden (state) aria-invalid (state) aria-label aria-labelledby aria-live aria-owns aria-relevant
Name From:	author

---

landmark (abstract role) §

A perceivable section containing content that is relevant to a specific, author-specified purpose and sufficiently important that users will likely want to be able to navigate to the section easily and to have it listed in a summary of the page. Such a page summary could be generated dynamically by a user agent or assistive technology.

Authors designate the purpose of the content by assigning a role that is a subclass of the landmark role and, when needed, by providing a brief, descriptive label.

Elements with a role that is a subclass of the landmark role are known as landmark regions or navigational landmark regions. Assistive technologies SHOULD enable users to quickly navigate to landmark regions. Mainstream user agents MAY enable users to quickly navigate to landmark regions.

NOTE

landmark is an abstract role used for the ontology. Authors should not use this role in content.

Characteristics:

Characteristic	Value
Is Abstract:	True
Superclass Role:	section
Subclass Roles:	application banner complementary contentinfo form main navigation region search
Inherited States and Properties:	aria-atomic aria-busy (state) aria-controls aria-current (state) aria-describedat aria-describedby aria-disabled (state) aria-dropeffect aria-expanded (state) aria-flowto aria-grabbed (state) aria-haspopup aria-hidden (state) aria-invalid (state) aria-label aria-labelledby aria-live aria-owns aria-relevant
Name From:	author
Accessible Name Required:	False

link (role) §

An interactive reference to an internal or external resource that, when activated, causes the user agent to navigate to that resource. See related button.

If this is a native link in the host language (such as an HTML anchor with an href value value), activating the link causes the user agent to navigate to that resource. If this is a simulated link, the web application author is responsible for managing navigation.

NOTE

If pressing the link triggers an action but does not change browser focus or page location, authors are advised to consider using the button role instead of the link role.

Characteristics:

Characteristic	Value
Superclass Role:	command
Related Concepts:	HTML link
Supported States and Properties:	aria-expanded
Inherited States and Properties:	aria-atomic aria-busy (state) aria-controls aria-current (state) aria-describedat aria-describedby aria-disabled (state) aria-dropeffect aria-flowto aria-grabbed (state) aria-haspopup aria-hidden (state) aria-invalid (state) aria-label

	aria-labelledby aria-live aria-owns aria-relevant
Name From:	contents author
Accessible Name Required:	True

list (role) §

A group of non-interactive list items. See related listbox.

Lists contain children whose role is listitem, or elements whose role is group which in turn contains children whose role is listitem.

Characteristics:

Characteristic	Value
Superclass Role:	section
Subclass Roles:	directory listbox menu
Base Concept:	HTML ul HTML ol
Required Owned Elements:	group → listitem listitem
Inherited States and Properties:	aria-atomic aria-busy (state) aria-controls aria-current (state) aria-describedat aria-describedby aria-disabled (state) aria-dropeffect aria-expanded (state) aria-flowto aria-grabbed (state) aria-haspopup aria-hidden (state) aria-invalid (state) aria-label aria-labelledby aria-live aria-owns aria-relevant
Name From:	author

listbox (role) §

A widget that allows the user to select one or more items from a list of choices. See related combobox and list.

Items within the list are static and, unlike standard HTML select elements, may contain images. List boxes contain children whose role is option.

To be keyboard accessible, authors SHOULD manage focus of descendants for all instances of this role, as described in Managing Focus.

> NOTE
>
> Elements with the role listbox have an implicit aria-orientation value of vertical.

Characteristics:

Characteristic	Value
Superclass Role:	list select
Related Concepts:	HTML select XForms select
Required Owned Elements:	option
Supported States and Properties:	aria-multiselectable aria-required
Inherited States and Properties:	aria-activedescendant aria-atomic aria-busy (state) aria-controls

	aria-current (state) aria-describedat aria-describedby aria-disabled (state) aria-dropeffect aria-expanded (state) aria-flowto aria-grabbed (state) aria-haspopup aria-hidden (state) aria-invalid (state) aria-label aria-labelledby aria-live aria-orientation aria-owns aria-relevant
Name From:	author
Accessible Name Required:	True
Implicit Value for Role:	Default for aria-orientation is vertical.

---

listitem (role) §

A single item in a list or directory.

Authors MUST ensure elements with role listitem are contained in, or owned by, an element with the role list or group.

Characteristics:

Characteristic	Value
Superclass Role:	section
Subclass Roles:	treeitem
Base Concept:	HTML li
Related Concepts:	XForms item
Required Context Role:	group list
Supported States and Properties:	aria-level aria-posinset aria-setsize
Inherited States and Properties:	aria-atomic aria-busy (state) aria-controls aria-current (state) aria-describedat aria-describedby aria-disabled (state) aria-dropeffect aria-expanded (state) aria-flowto aria-grabbed (state) aria-haspopup aria-hidden (state) aria-invalid (state) aria-label aria-labelledby aria-live aria-owns aria-relevant
Name From:	contents author
Accessible Name Required:	True

---

log (role) §

A type of live region where new information is added in meaningful order and old information may disappear. See related marquee.

Examples include chat logs, messaging history, game log, or an error log. In contrast to other live regions, in this role there is a relationship between the arrival of new items in the log and the reading order. The log contains a meaningful sequence and new information is added only to the end of the log, not at arbitrary points.

> NOTE
>
> Elements with the role log have an implicit aria-live value of polite.

Characteristics:

Characteristic	Value
Superclass Role:	section
Inherited States and Properties:	aria-atomic aria-busy (state) aria-controls aria-current (state) aria-describedat aria-describedby aria-disabled (state) aria-dropeffect aria-expanded (state) aria-flowto aria-grabbed (state) aria-haspopup aria-hidden (state) aria-invalid (state) aria-label aria-labelledby aria-live aria-owns aria-relevant
Name From:	author
Accessible Name Required:	True
Implicit Value for Role:	Default for aria-live is polite.

---

main (role) §

The main content of a document.

This marks the content that is directly related to or expands upon the central topic of the document. The main role is a non-obtrusive alternative for "skip to main content" links, where the navigation option to go to the main content (or other landmarks) is provided by the user agent through a dialog or by assistive technologies.

User agents SHOULD treat elements with the role of main as navigational landmarks.

Within any document or application, the author SHOULD mark no more than one element with the main role.

> NOTE
>
> Because document and application elements can be nested in the DOM, they may have multiple main elements as DOM descendants, assuming each of those is associated with different document nodes, either by a DOM nesting (e.g., document within document) or by use of the aria-owns attribute.

Characteristics:

Characteristic	Value
Superclass Role:	landmark
Inherited States and Properties:	aria-atomic aria-busy (state) aria-controls aria-current (state) aria-describedat aria-describedby aria-disabled (state) aria-dropeffect aria-expanded (state) aria-flowto aria-grabbed (state) aria-haspopup aria-hidden (state) aria-invalid (state) aria-label aria-labelledby aria-live aria-owns aria-relevant
Name From:	author

---

marquee (role) §

A type of live region where non-essential information changes frequently. See related log.

Common usages of marquee include stock tickers and ad banners. The primary difference between a marquee and a log is that

logs usually have a meaningful order or sequence of important content changes.

> NOTE
>
> Elements with the role marquee have an implicit aria-live value of off.

Characteristics:

Characteristic	Value
Superclass Role:	section
Inherited States and Properties:	aria-atomic aria-busy (state) aria-controls aria-current (state) aria-describedat aria-describedby aria-disabled (state) aria-dropeffect aria-expanded (state) aria-flowto aria-grabbed (state) aria-haspopup aria-hidden (state) aria-invalid (state) aria-label aria-labelledby aria-live aria-owns aria-relevant
Name From:	author
Accessible Name Required:	True

---

math (role) §

Content that represents a mathematical expression.

Content with the role math is intended to be marked up in an accessible format such as MathML [MathML], or with another type of textual representation such as TeX or LaTeX, which can be converted to an accessible format by native browser implementations or a polyfill library.

While it is not ideal to use an image of a mathematical expression, there exists a significant amount of legacy content where images are used to represent mathematical expressions. Authors SHOULD ensure that images of math are labeled by text that describes the mathematical expression as it might be spoken.

> NOTE
>
> Browsers that support native implementations of MathML are able to provide a more robust, accessible math experience than can be accomplished with plain text approximations of math. Some rendering engines have close integration with screen readers that allow spacial touch exploration of the formula and refreshable braille display output in the Nemeth Braille format. This level of integration is not supported with images of mathematical formulas, even if the author provides a plain text approximation.
>
> At the time of this writing, some mainstream browsers do not support MathML natively, and must be retrofit using a JavaScript polyfill library. When authoring math content, use native MathML wherever possible, and test thoroughly. Use a polyfill library or provide a fallback image with a text alternative approximation if necessary.

MathML Example with Embedded TeX Annotation §

EXAMPLE 6

```
<!-- Note: Use a JavaScript polyfill library to ensure
     this renders in user agents that do not support MathML. -->
<!-- The math element has an implicit role="math". -->
<math xmlns="http://www.w3.org/1998/Math/MathML">
  <mrow>
    <mi>x</mi>
    <mo>=</mo>
    <mfrac>
      <mrow>
        <mo form="prefix">−</mo>
        <mi>b</mi>
        <mo>±</mo>
        <msqrt>
          <msup>
            <mi>b</mi>
            <mn>2</mn>
          </msup>
          <mo>−</mo>
          <mn>4</mn>
```

```
          <mo>&#x2062;<!-- &InvisibleTimes; --></mo>
          <mi>a</mi>
          <mo>&#x2062;<!-- &InvisibleTimes; --></mo>
          <mi>c</mi>
        </msqrt>
      </mrow>
      <mrow>
        <mn>2</mn>
        <mo>&#x2062;<!-- &InvisibleTimes; --></mo>
        <mi>a</mi>
      </mrow>
    </mfrac>
  </mrow>
  <annotation encoding="TeX">
    x=\frac{-b\pm\sqrt{b^2-4ac}}{2a}
  </annotation>
</math>
```

Plain HTML or Polyfill DOM Result of the MathML Quadratic Formula §

If a rendering engine does not support a native math format such as MathML, authors MAY use JavaScript to downgrade the content to a format the browser can display, such as this HTML image using a data URI and plain text alternative.

EXAMPLE 7

```
<img role="math" src="..." alt="x=□-b±√□b²-4ac□□÷2a">
```

EDITOR'S NOTE

Editor's note: Might need an RFC-2119 "should" requirement here to encourage AT to speak math approximations with high punctuation verbosity. Otherwise ambiguous characters like a forward slash (/) may not be spoken even when intended to be used interchangeably with the division sign character (÷).

Characteristics:

Characteristic	Value
Superclass Role:	section
Inherited States and Properties:	aria-atomic aria-busy (state) aria-controls aria-current (state) aria-describedat aria-describedby aria-disabled (state) aria-dropeffect aria-expanded (state) aria-flowto aria-grabbed (state) aria-haspopup aria-hidden (state) aria-invalid (state) aria-label aria-labelledby aria-live aria-owns aria-relevant
Name From:	author
Children Presentational:	True

---

menu (role) §

A type of widget that offers a list of choices to the user.

A menu is often a list of common actions or functions that the user can invoke. The menu role is appropriate when a list of menu items is presented in a manner similar to a menu on a desktop application.

To be keyboard accessible, authors SHOULD manage focus of descendants for all instances of this role, as described in Managing Focus.

NOTE

Elements with the role menu have an implicit aria-orientation value of vertical.

Characteristics:

Characteristic	Value

Superclass Role:	list select
Subclass Roles:	menubar
Related Concepts:	DTB sidebar XForms select JAPI MENU
Required Owned Elements:	group → menuitemradio menuitem menuitemcheckbox menuitemradio
Inherited States and Properties:	aria-activedescendant aria-atomic aria-busy (state) aria-controls aria-current (state) aria-describedat aria-describedby aria-disabled (state) aria-dropeffect aria-expanded (state) aria-flowto aria-grabbed (state) aria-haspopup aria-hidden (state) aria-invalid (state) aria-label aria-labelledby aria-live aria-orientation aria-owns aria-relevant
Name From:	author
Implicit Value for Role:	Default for aria-orientation is vertical.

menubar (role) §

A presentation of menu that usually remains visible and is usually presented horizontally.

The menubar role is used to create a menu bar similar to those found in Windows, Mac, and Gnome desktop applications. A menu bar is used to create a consistent set of frequently used commands. Authors SHOULD ensure that menubar interaction is similar to the typical menu bar interaction in a desktop graphical user interface.

To be keyboard accessible, authors SHOULD manage focus of descendants for all instances of this role, as described in Managing Focus.

> NOTE
>
> Elements with the role menubar have an implicit aria-orientation value of horizontal.

Characteristics:

Characteristic	Value
Superclass Role:	menu
Related Concepts:	toolbar
Required Owned Elements:	group → menuitemradio menuitem menuitemcheckbox menuitemradio
Inherited States and Properties:	aria-activedescendant aria-atomic aria-busy (state) aria-controls aria-current (state) aria-describedat aria-describedby aria-disabled (state) aria-dropeffect aria-expanded (state) aria-flowto aria-grabbed (state) aria-haspopup aria-hidden (state) aria-invalid (state) aria-label aria-labelledby aria-live aria-orientation

	aria-owns aria-relevant
Name From:	author
Implicit Value for Role:	Default for aria-orientation is horizontal.

menuitem (role) §

An option in a set of choices contained by a menu or menubar.

Authors MAY disable a menu item with the aria-disabled attribute. If the menu item has its aria-haspopup attribute set to true, it indicates that the menu item may be used to launch a sub-level menu, and authors SHOULD display a new sub-level menu when the menu item is activated.

Authors MUST ensure that menu items are owned by an element with role menu or menubar in order to identify that they are related widgets. Authors MAY separate menu items into sets by use of a separator or an element with an equivalent role from the native markup language.

Characteristics:

Characteristic	Value
Superclass Role:	command
Subclass Roles:	menuitemcheckbox
Related Concepts:	JAPI MENU ITEM listitem option
Required Context Role:	group menu menubar
Inherited States and Properties:	aria-atomic aria-busy (state) aria-controls aria-current (state) aria-describedat aria-describedby aria-disabled (state) aria-dropeffect aria-flowto aria-grabbed (state) aria-haspopup aria-hidden (state) aria-invalid (state) aria-label aria-labelledby aria-live aria-owns aria-relevant
Name From:	contents author
Accessible Name Required:	True

menuitemcheckbox (role) §

A menuitem with a checkable state whose possible values are true, false, or mixed.

The aria-checked attribute of a menuitemcheckbox indicates whether the menu item is checked (true), unchecked (false), or represents a sub-level menu of other menu items that have a mixture of checked and unchecked values (mixed).

Authors MUST ensure that menu item checkboxes are owned by an element with role menu or menubar in order to identify that they are related widgets. Authors MAY separate menu items into sets by use of a separator or an element with an equivalent role from the native markup language.

Characteristics:

Characteristic	Value
Superclass Role:	checkbox menuitem
Subclass Roles:	menuitemradio
Related Concepts:	menuitem
Required Context Role:	menu menubar
Inherited States and Properties:	aria-atomic aria-busy (state) aria-checked (state) aria-controls aria-current (state) aria-describedat aria-describedby

	aria-disabled (state) aria-dropeffect aria-flowto aria-grabbed (state) aria-haspopup aria-hidden (state) aria-invalid (state) aria-label aria-labelledby aria-live aria-owns aria-relevant
Name From:	contents author
Accessible Name Required:	True
Implicit Value for Role:	Default for aria-checked is false.

menuitemradio (role) §

A checkable menuitem in a set of elements with the same role, only one of which can be checked at a time.

Authors SHOULD enforce that only one menuitemradio in a group can be checked at the same time. When one item in the group is checked, the previously checked item becomes unchecked (its aria-checked attribute becomes false).

Authors MUST ensure that menu item radios are owned by an element with role group, menu, or menubar in order to identify that they are related widgets. Authors MAY separate menu items into sets by use of a separator or an element with an equivalent role from the native markup language.

If a menu or menubar contains more than one group of menuitemradio elements, or if the menu contains one group and other, unrelated menu items, authors SHOULD nest each set of related menuitemradio elements in an element using the group role, and authors SHOULD delimit the group from other menu items with an element using the separator role.

Characteristics:

Characteristic	Value
Superclass Role:	menuitemcheckbox (see structure) radio
Related Concepts:	menuitem
Required Context Role:	group menu menubar
Inherited States and Properties:	aria-atomic aria-busy (state) aria-checked (state) aria-controls aria-current (state) aria-describedat aria-describedby aria-disabled (state) aria-dropeffect aria-flowto aria-grabbed (state) aria-haspopup aria-hidden (state) aria-invalid (state) aria-label aria-labelledby aria-live aria-owns aria-posinset aria-relevant aria-setsize
Name From:	contents author
Accessible Name Required:	True
Implicit Value for Role:	Default for aria-checked is false.

navigation (role) §

A collection of navigational elements (usually links) for navigating the document or related documents.

User agents SHOULD treat elements with the role of navigation as navigational landmarks.

Characteristics:

Characteristic	Value
Superclass Role:	landmark
Related Concepts:	nav element

Inherited States and Properties:	aria-atomic aria-busy (state) aria-controls aria-current (state) aria-describedat aria-describedby aria-disabled (state) aria-dropeffect aria-expanded (state) aria-flowto aria-grabbed (state) aria-haspopup aria-hidden (state) aria-invalid (state) aria-label aria-labelledby aria-live aria-owns aria-relevant
Name From:	author

none (role) §

[ARIA 1.1] An element whose implicit native role semantics will not be mapped to the accessibility API. See synonym presentation [ARIA 1.0].

EDITOR'S NOTE

Editorial Note regarding the ARIA 1.1 none role.

In ARIA 1.1, the working group plans to introduce none as a synonym to the presentation role, due to author confusion surrounding the intended meaning of the word "presentation" or "presentational." Many individuals erroneously consider role="presentation" to be synonymous with aria-hidden="true", and we believe role="none" conveys the actual meaning more unambiguously.

Until implementations include sufficient support for role="none", web authors are advised to use the presentation role alone role="presentation" or redundantly as a fallback to the none role role="none presentation".

note (role) §

A section whose content is parenthetic or ancillary to the main content of the resource.

Characteristics:

Characteristic	Value
Superclass Role:	section
Inherited States and Properties:	aria-atomic aria-busy (state) aria-controls aria-current (state) aria-describedat aria-describedby aria-disabled (state) aria-dropeffect aria-expanded (state) aria-flowto aria-grabbed (state) aria-haspopup aria-hidden (state) aria-invalid (state) aria-label aria-labelledby aria-live aria-owns aria-relevant
Name From:	author

option (role) §

A selectable item in a select list.

Authors MUST ensure elements with role option are contained in, or owned by, an element with the role listbox. Options not associated with a listbox might not be correctly mapped to an accessibility API.

NOTE

Elements with the role option have an implicit aria-selected value of false.

Characteristics:

Characteristic	Value
Superclass Role:	input
Subclass Roles:	treeitem
Base Concept:	HTML option
Related Concepts:	listitem XForms item
Required Context Role:	listbox
Required States and Properties:	aria-selected
Supported States and Properties:	aria-checked aria-posinset aria-setsize
Inherited States and Properties:	aria-atomic aria-busy (state) aria-controls aria-current (state) aria-describedat aria-describedby aria-disabled (state) aria-dropeffect aria-flowto aria-grabbed (state) aria-haspopup aria-hidden (state) aria-invalid (state) aria-label aria-labelledby aria-live aria-owns aria-relevant
Name From:	contents author
Accessible Name Required:	True
Implicit Value for Role:	Default for aria-selected is false.

presentation (role) §

An element whose implicit native role semantics will not be mapped to the accessibility API. See synonym none [ARIA 1.1].

EDITOR'S NOTE

Editorial Note regarding the ARIA 1.1 none role.

In ARIA 1.1, the working group plans to introduce none as a synonym to the presentation role, due to author confusion surrounding the intended meaning of the word "presentation" or "presentational." Many individuals erroneously consider role="presentation" to be synonymous with aria-hidden="true", and we believe role="none" conveys the actual meaning more unambiguously.

Until implementations include sufficient support for role="none", web authors are advised to use the presentation role alone role="presentation" or redundantly as a fallback to the none role role="none presentation".

The intended use is when an element is used to change the look of the page but does not have all the functional, interactive, or structural relevance implied by the element type, or may be used to provide for an accessible fallback in older browsers that do not support WAI-ARIA.

Example use cases:

- An element whose content is completely presentational (like a spacer image, decorative graphic, or clearing element);
- An image that is in a container with the img role and where the full text alternative is available and is marked up with aria-labelledby and (if needed) aria-describedby;
- An element used as an additional markup "hook" for CSS; or
- A layout table and/or any of its associated rows, cells, etc.

For any element with a role of presentation and which is not focusable, the user agent MUST NOT expose the implicit native semantics of the element (the role and its states and properties) to accessibility APIs. However, the user agent MUST expose content and descendant elements that do not have an explicit or inherited role of presentation. Thus, the presentation role causes a given element to be treated as having no role or to be removed from the accessibility tree, but does not cause the content contained within the element to be removed from the accessibility tree.

For example, according to an accessibility API, the following markup elements would appear to have identical role semantics (no role) and identical content.

EXAMPLE 8

```
<!-- 1. [role="presentation"] negates the implicit 'heading' role semantics but does not affect the contents. -->
<h1 role="presentation"> Sample Content </h1>

<!-- 2. There is no implicit role for span, so only the contents are exposed. -->
<span> Sample Content </span>

<!-- 3. Depending on styling and other factors, this role declaration is redundant in some implementations. -->
<span role="presentation"> Sample Content </span>

<!-- 4. In all cases, the element contents are exposed to accessibility APIs without any implied role semantics. -->
<!-- <> --> Sample Content <!-- </> -->
```

The presentation role is used on an element that has implicit native semantics, meaning that there is a default accessibility API role for the element. Some elements are only complete when additional descendant elements are provided. For example, in HTML, table elements (matching the grid role) require tr descendants (the row role), which in turn require th or td children (the gridcell, columnheader, rowheader roles). Similarly, lists require list item children. The descendant elements that complete the semantics of an element are described in WAI-ARIA as required owned elements.

When an explicit or inherited role of presentation is applied to an element with the implicit semantic of a WAI-ARIA role that has required owned elements, in addition to the element with the explicit role of presentation, the user agent MUST apply an inherited role of presentation to any owned elements that do not have an explicit role defined. Also, when an explicit or inherited role of presentation is applied to a host language element which has required children as defined by the host language specification, in addition to the element with the explicit role of presentation, the user agent MUST apply an inherited role of presentation to any required children that do not have an explicit role defined. For any element with an explicit or inherited role of presentation and which is not focusable, user agents MUST ignore role-specific WAI-ARIA states and properties for that element. For example, in HTML, a ul or ol element with a role of presentation will have the implicit native semantics of its li elements removed because the list role to which the ul or ol corresponds has a required owned element of listitem. Likewise, although an HTML table element does not have an implicit native semantic role corresponding directly to a WAI-ARIA role, the implicit native semantics of its thead/tbody/tfoot/tr/th/td descendants will also be removed, because the HTML specification indicates that these are required structural descendants of the table element. Explicit roles on a descendant or owned element override the inherited role of presentation, and cause the owned element to behave as any other element with an explicit role. If the action of exposing the implicit role causes the accessibility tree to be malformed, the expected results are undefined and the user agent MAY resort to an internal recovery mechanism to repair the accessibility tree.

NOTE

Only the implicit native semantics of elements that correspond to WAI-ARIA required owned elements are removed. All other content remains intact, including nested tables or lists, unless those elements also have a explicit role of presentation applied.

For example, according to an accessibility API, the following markup elements would appear to have identical role semantics (no roles) and identical content.

EXAMPLE 9

```
<!-- 1. [role="presentation"] negates the implicit 'list' and 'listitem' role semantics but does not affect the contents. -->
<ul role="presentation">
  <li> Sample Content </li>
  <li> More Sample Content </li>
</ul>

<!-- 2. There is no implicit role for "foo", so only the contents are exposed. -->
<foo>
  <foo> Sample Content </foo>
  <foo> More Sample Content </foo>
</foo>
```

NOTE

There are other WAI-ARIA roles with required children for which this situation is applicable (e.g., radiogroups and listboxes), but tables and lists are the most common real-world cases in which the presentation inheritance is likely to apply.

For any element with an explicit or inherited role of presentation, user agents MUST apply an inherited role of presentation to all host-language-specific labeling elements for the presentational element. For example, a table element with a role of presentation will have the implicit native semantics of its caption element removed, because the caption is merely a label for the presentational table.

For any element with an explicit or inherited role of presentation, user agents MUST ignore any non-global, role-specific WAI-ARIA states and properties. However, the user agent MUST always expose global WAI-ARIA states and properties to accessibility APIs, even if an element has an explicit or inherited role of presentation.

For example, aria-hidden is a global attribute and would always be applied; aria-level is not a global attribute and would therefore only apply if the element was not in a presentational state.

EXAMPLE 10

```
<!-- 1. [role="presentation"] negates the implicit 'heading' role semantics but does not affect the global hidden state. -->
<h1 role="presentation" aria-hidden="true"> Sample Content </h1>

<!-- 1. [role="presentation"] negates the both the implicit 'heading' and the non-global level. -->
<h1 role="presentation" aria-level="2"> Sample Content </h1>
```

If an element with a role of presentation is focusable, user agents MUST ignore the normal effect of the role and expose the element with implicit native semantics, in order to ensure that the element is both understandable and operable. Authors SHOULD NOT provide meaningful alternative text (for example, use alt="" in HTML4) when the presentation role is applied to an image.

In the following code sample, the containing img and is appropriately labeled by the caption paragraph. In this example the img element can be marked as presentation because the role and the text alternatives are provided by the containing element.

EXAMPLE 11

```
<div role="img" aria-labelledby="caption">
  <img src="example.png" role="presentation" alt="">
  <p id="caption">A visible text caption labeling the image.</p>
</div>
```

In the following code sample, because the anchor (HTML a element) is acting as the treeitem, the list item (HTML li element) is assigned an explicit WAI-ARIA role of presentation to override the user agent's implicit native semantics for list items.

EXAMPLE 12

```
<ul role="tree">
  <li role="presentation">
        <a role="treeitem" aria-expanded="true">An expanded tree node</a>
  </li>
  …
</ul>
```

Characteristics:

Characteristic	Value
Superclass Role:	structure
Inherited States and Properties:	aria-atomic aria-busy (state) aria-controls aria-current (state) aria-describedat aria-describedby aria-disabled (state) aria-dropeffect aria-flowto aria-grabbed (state) aria-haspopup aria-hidden (state) aria-invalid (state) aria-label aria-labelledby aria-live aria-owns aria-relevant
Name From:	author (if role discarded by error conditions)

progressbar (role) §

An element that displays the progress status for tasks that take a long time.

A progressbar indicates that the user's request has been received and the application is making progress toward completing the requested action. The author SHOULD supply values for aria-valuenow, aria-valuemin, and aria-valuemax, unless the value is indeterminate, in which case the aauthor SHOULD omit the aria-valuenow attribute. Authors SHOULD update these values when the visual progress indicator is updated. If the progressbar is describing the loading progress of a particular region of a page, the author SHOULD use aria-describedby to point to the status, and set the aria-busy attribute to true on the region until it is finished loading. It is not possible for the user to alter the value of a progressbar because it is always readonly.

NOTE

Assistive technologies generally will render the value of aria-valuenow as a percent of a tange between the value of aria-valuemin and aria-valuemax, unless aria-valuetext is specified. It is best to set the values for aria-valuemin, aria-valuemax, and aria-valuenow in a manner that is appropriate for this calculation.

NOTE

Elements with the role progressbar have an implicit aria-readonly value of true.

Characteristics:

Characteristic	Value
Superclass Role:	range
Related Concepts:	status
Inherited States and Properties:	aria-atomic aria-busy (state) aria-controls aria-current (state) aria-describedat aria-describedby aria-disabled (state) aria-dropeffect aria-expanded (state) aria-flowto aria-grabbed (state) aria-haspopup aria-hidden (state) aria-invalid (state) aria-label aria-labelledby aria-live aria-owns aria-relevant aria-valuemax aria-valuemin aria-valuenow aria-valuetext
Name From:	author
Accessible Name Required:	True
Children Presentational:	True
Implicit Value for Role:	Default for aria-readonly is true.

radio (role) §

A checkable input in a group of elements with the same role, only one of which can be checked at a time.

Authors SHOULD ensure that elements with role radio are explicitly grouped in order to indicate which ones affect the same value. This is achieved by enclosing the radio elements in an element with role radiogroup. If it is not possible to make the radio buttons DOM children of the radiogroup, authors SHOULD use the aria-owns attribute on the radiogroup element to indicate the relationship to its children.

Characteristics:

Characteristic	Value
Superclass Role:	checkbox
Subclass Roles:	menuitemradio
Related Concepts:	HTML input[type="radio"]
Required States and Properties:	aria-checked
Supported States and Properties:	aria-posinset aria-setsize
Inherited States and Properties:	aria-atomic aria-busy (state) aria-checked (state) aria-controls aria-current (state) aria-describedat aria-describedby aria-disabled (state) aria-dropeffect aria-flowto aria-grabbed (state) aria-haspopup aria-hidden (state) aria-invalid (state) aria-label aria-labelledby aria-live aria-owns aria-relevant
Name From:	contents author

Accessible Name Required:	True
Implicit Value for Role:	Default for aria-checked is false.

radiogroup (role) §

A group of radio buttons.

A radiogroup is a type of select list that can only have a single entry checked at any one time. Authors SHOULD enforce that only one radio button in a group can be checked at the same time. When one item in the group is checked, the previously checked item becomes unchecked (its aria-checked attribute becomes false).

Characteristics:

Characteristic	Value
Superclass Role:	select
Related Concepts:	list
Required Owned Elements:	radio
Supported States and Properties:	aria-required
Inherited States and Properties:	aria-activedescendant aria-atomic aria-busy (state) aria-controls aria-current (state) aria-describedat aria-describedby aria-disabled (state) aria-dropeffect aria-expanded (state) aria-flowto aria-grabbed (state) aria-haspopup aria-hidden (state) aria-invalid (state) aria-label aria-labelledby aria-live aria-orientation aria-owns aria-relevant
Name From:	author
Accessible Name Required:	True

range (abstract role) §

An input representing a range of values that can be set by the user.

> NOTE
>
> range is an abstract role used for the ontology. Authors should not use this role in content.

Characteristics:

Characteristic	Value
Is Abstract:	True
Superclass Role:	widget
Subclass Roles:	progressbar scrollbar slider spinbutton
Supported States and Properties:	aria-valuemax aria-valuemin aria-valuenow aria-valuetext
Inherited States and Properties:	aria-atomic aria-busy (state) aria-controls aria-current (state) aria-describedat aria-describedby aria-disabled (state) aria-dropeffect aria-flowto aria-grabbed (state) aria-haspopup

	aria-hidden (state) aria-invalid (state) aria-label aria-labelledby aria-live aria-owns aria-relevant
Name From:	author

region (role) §

A perceivable section containing content that is relevant to a specific, author-specified purpose and sufficiently important that users will likely want to be able to navigate to the section easily and to have it listed in a summary of the page. Such a page summary could be generated dynamically by a user agent or assistive technology.

Authors SHOULD limit use of the region role to sections containing content with a purpose that is not accurately described by one of the other landmark roles, such as main, complementary, or navigation.

Authors MUST give each element with role region a brief label that describes the purpose of the content in the region. Authors SHOULD reference a visible label with aria-labelledby if a visible label is present. Authors SHOULD include the label inside of a heading whenever possible. The heading MAY be an instance of the standard host language heading element or an instance of an element with role heading.

Assistive technologies SHOULD enable users to quickly navigate to elements with role region. Mainstream user agents MAY enable users to quickly navigate to elements with role region.

Characteristics:

Characteristic	Value
Superclass Role:	landmark
Related Concepts:	HTML Frame Device Independence Glossary perceivable unit section
Inherited States and Properties:	aria-atomic aria-busy (state) aria-controls aria-current (state) aria-describedat aria-describedby aria-disabled (state) aria-dropeffect aria-expanded (state) aria-flowto aria-grabbed (state) aria-haspopup aria-hidden (state) aria-invalid (state) aria-label aria-labelledby aria-live aria-owns aria-relevant
Name From:	author
Accessible Name Required:	True

roletype (abstract role) §

The base role from which all other roles in this taxonomy inherit.

Properties of this role describe the structural and functional purpose of objects that are assigned this role (known in RDF terms as "instances"). A role is a concept that can be used to understand and operate instances.

> NOTE
>
> roletype is an abstract role used for the ontology. Authors should not use this role in content.

Characteristics:

Characteristic	Value
Is Abstract:	True
Subclass Roles:	structure widget window
Related Concepts:	XHTML role HTML link (rel & rev) Dublin Core type

Supported States and Properties:	aria-atomic aria-busy (state) aria-controls aria-current (state) aria-describedat aria-describedby aria-disabled (state) aria-dropeffect aria-flowto aria-grabbed (state) aria-haspopup aria-hidden (state) aria-invalid (state) aria-label aria-labelledby aria-live aria-owns aria-relevant
Name From:	n/a

row (role) §

A row of cells in a grid.

Rows contain gridcell elements, and thus serve to organize the grid.

In a treegrid, authors MAY mark rows as expandable, using the aria-expanded attribute to indicate the present status. This is not the case for an ordinary grid, in which the aria-expanded attribute is not present.

Authors MUST ensure elements with role row are contained in, or owned by, and element with the role grid, rowgroup, or treegrid.

Characteristics:

Characteristic	Value
Superclass Role:	group widget
Base Concept:	HTML tr
Required Context Role:	grid rowgroup treegrid
Required Owned Elements:	columnheader gridcell rowheader
Supported States and Properties:	aria-colindex aria-level aria-rowindex aria-selected
Inherited States and Properties:	aria-activedescendant aria-atomic aria-busy (state) aria-controls aria-current (state) aria-describedat aria-describedby aria-disabled (state) aria-dropeffect aria-expanded (state) aria-flowto aria-grabbed (state) aria-haspopup aria-hidden (state) aria-invalid (state) aria-label aria-labelledby aria-live aria-owns aria-relevant
Name From:	contents author

rowgroup (role) §

A structure containing one or more row elements in a grid.

The rowgroup role establishes a relationship between owned row elements. It is a structural equivalent to the thead, tfoot, and tbody elements in an HTML table element.

Authors MUST ensure elements with role rowgroup are contained in, or owned by, an element with the role grid.

NOTE

The rowgroup role exists, in part, to support role symmetry in HTML, and allows for the propagation of presentation inheritance on HTML table elements with an explicit presentation role applied.

NOTE

This role does not differentiate between types of row groups (e.g., thead vs. tbody), but an issue has been raised for WAI-ARIA 2.0.

Characteristics:

Characteristic	Value
Superclass Role:	structure
Base Concept:	HTML thead, tfoot, and tbody
Required Context Role:	grid treegrid
Required Owned Elements:	row
Inherited States and Properties:	aria-atomic aria-busy (state) aria-controls aria-current (state) aria-describedat aria-describedby aria-disabled (state) aria-dropeffect aria-flowto aria-grabbed (state) aria-haspopup aria-hidden (state) aria-invalid (state) aria-label aria-labelledby aria-live aria-owns aria-relevant
Name From:	contents author

rowheader (role) §

A cell containing header information for a row in a grid.

Rowheader can be used as a row header in a table or grid. The rowheader establishes a relationship between it and all cells in the corresponding row. It is a structural equivalent to setting scope="row" on an HTML th element.

Authors MUST ensure elements with role rowheader are contained in, or owned by, an element with the role grid.

Applying the aria-selected state on a rowheader MUST not cause the user agent to automatically propogate the aria-selected state to all the cells in the corresponding row. An author MAY choose to propogate selection in this manner depending on the specific application.

Characteristics:

Characteristic	Value
Superclass Role:	gridcell sectionhead widget
Base Concept:	HTML th[scope="row"]
Required Context Role:	row
Supported States and Properties:	aria-sort
Inherited States and Properties:	aria-atomic aria-busy (state) aria-colindex aria-colspan aria-controls aria-current (state) aria-describedat aria-describedby aria-disabled (state) aria-dropeffect aria-expanded (state)

	aria-flowto aria-grabbed (state) aria-haspopup aria-hidden (state) aria-invalid (state) aria-label aria-labelledby aria-live aria-owns aria-readonly aria-relevant aria-required aria-rowindex aria-rowspan aria-selected (state)
Name From:	contents author
Accessible Name Required:	True

search (role) §

A landmark region that contains a collection of items and objects that, as a whole, combine to create a search facility. See related form and searchbox.

A search region may be a mix of host language form controls, scripted controls, and hyperlinks.

User agents SHOULD treat elements with the role of search as navigational landmarks.

Characteristics:

Characteristic	Value
Superclass Role:	landmark
Inherited States and Properties:	aria-atomic aria-busy (state) aria-controls aria-current (state) aria-describedat aria-describedby aria-disabled (state) aria-dropeffect aria-expanded (state) aria-flowto aria-grabbed (state) aria-haspopup aria-hidden (state) aria-invalid (state) aria-label aria-labelledby aria-live aria-owns aria-relevant
Name From:	author

searchbox (role) §

[ARIA 1.1] A type of textbox intended for specifying search criteria. See related textbox and search.

Characteristics:

Characteristic	Value
Superclass Role:	textbox
Base Concept:	HTML input[type="search"]
Inherited States and Properties:	aria-activedescendant aria-atomic aria-autocomplete aria-busy (state) aria-controls aria-current (state) aria-describedat aria-describedby aria-disabled (state) aria-dropeffect aria-flowto aria-grabbed (state) aria-haspopup aria-hidden (state) aria-invalid (state)

	aria-label aria-labelledby aria-live aria-multiline aria-owns aria-placeholder aria-readonly aria-relevant aria-required
Name From:	author
Accessible Name Required:	True

section (abstract role) §

A renderable structural containment unit in a document or application.

> NOTE
>
> section is an abstract role used for the ontology. Authors should not use this role in content.

Characteristics:

Characteristic	Value
Is Abstract:	True
Superclass Role:	structure
Subclass Roles:	alert definition grid gridcell group img landmark list listitem log marquee math note status tabpanel tooltip
Related Concepts:	DTB frontmatter DTB level SMIL par
Supported States and Properties:	aria-expanded
Inherited States and Properties:	aria-atomic aria-busy (state) aria-controls aria-current (state) aria-describedat aria-describedby aria-disabled (state) aria-dropeffect aria-flowto aria-grabbed (state) aria-haspopup aria-hidden (state) aria-invalid (state) aria-label aria-labelledby aria-live aria-owns aria-relevant
Name From:	n/a

sectionhead (abstract role) §

A structure that labels or summarizes the topic of its related section.

> NOTE
>
> sectionhead is an abstract role used for the ontology. Authors should not use this role in content.

Characteristics:

Characteristic	Value
Is Abstract:	True
Superclass Role:	structure
Subclass Roles:	columnheader heading rowheader tab
Supported States and Properties:	aria-expanded
Inherited States and Properties:	aria-atomic aria-busy (state) aria-controls aria-current (state) aria-describedat aria-describedby aria-disabled (state) aria-dropeffect aria-flowto aria-grabbed (state) aria-haspopup aria-hidden (state) aria-invalid (state) aria-label aria-labelledby aria-live aria-owns aria-relevant
Name From:	contents author

select (abstract role) §

A form widget that allows the user to make selections from a set of choices.

> NOTE
>
> select is an abstract role used for the ontology. Authors should not use this role in content.

Characteristics:

Characteristic	Value
Is Abstract:	True
Superclass Role:	composite group input
Subclass Roles:	combobox listbox menu radiogroup tree
Supported States and Properties:	aria-orientation
Inherited States and Properties:	aria-activedescendant aria-atomic aria-busy (state) aria-controls aria-current (state) aria-describedat aria-describedby aria-disabled (state) aria-dropeffect aria-expanded (state) aria-flowto aria-grabbed (state) aria-haspopup aria-hidden (state) aria-invalid (state) aria-label aria-labelledby aria-live aria-owns aria-relevant
Name From:	author

separator (role) §

A divider that separates and distinguishes sections of content or groups of menuitems.

This is a visible separator between sections of content. For example, separators are found between groups of menu items in a menu or as the moveable separator between two regions in a split pane.

NOTE

Elements with the role separator have an implicit aria-orientation value of horizontal.

Characteristics:

Characteristic	Value
Superclass Role:	structure
Related Concepts:	HTML hr
Supported States and Properties:	aria-expanded aria-orientation
Inherited States and Properties:	aria-atomic aria-busy (state) aria-controls aria-current (state) aria-describedat aria-describedby aria-disabled (state) aria-dropeffect aria-flowto aria-grabbed (state) aria-haspopup aria-hidden (state) aria-invalid (state) aria-label aria-labelledby aria-live aria-owns aria-relevant
Name From:	author
Children Presentational:	True
Implicit Value for Role:	Default for aria-orientation is horizontal.

scrollbar (role) §

A graphical object that controls the scrolling of content within a viewing area, regardless of whether the content is fully displayed within the viewing area.

A scrollbar represents the current value and range of possible values via the size of the scrollbar and position of the thumb with respect to the visible range of the orientation (horizontal or vertical) it controls. Its orientation represents the orientation of the scrollbar and the scrolling effect on the viewing area controlled by the scrollbar. It is typically possible to add or subtract to the current value by using directional keys such as arrow keys.

Authors MUST set the aria-controls attribute on the scrollbar element to reference the scrollable area it controls.

NOTE

Elements with the role scrollbar have an implicit aria-orientation value of vertical.

NOTE

Assistive technologies generally will render the value of aria-valuenow as a percent of a tange between the value of aria-valuemin and aria-valuemax, unless aria-valuetext is specified. It is best to set the values for aria-valuemin, aria-valuemax, and aria-valuenow in a manner that is appropriate for this calculation.

Characteristics:

Characteristic	Value
Superclass Role:	input range
Required States and Properties:	aria-controls aria-orientation aria-valuemax aria-valuemin aria-valuenow
Inherited States and Properties:	aria-atomic aria-busy (state) aria-controls aria-current (state) aria-describedat

	aria-describedby aria-disabled (state) aria-dropeffect aria-flowto aria-grabbed (state) aria-haspopup aria-hidden (state) aria-invalid (state) aria-label aria-labelledby aria-live aria-owns aria-relevant aria-valuemax aria-valuemin aria-valuenow aria-valuetext
Name From:	author
Accessible Name Required:	False
Children Presentational:	True
Implicit Value for Role:	Default for aria-orientation is vertical.

slider (role) §

A user input where the user selects a value from within a given range.

A slider represents the current value and range of possible values via the size of the slider and position of the thumb. It is typically possible to add or subtract to the value by using directional keys such as arrow keys.

> NOTE
>
> Elements with the role slider have an implicit aria-orientation value of horizontal.

Characteristics:

Characteristic	Value
Superclass Role:	input range
Required States and Properties:	aria-valuemax aria-valuemin aria-valuenow
Supported States and Properties:	aria-orientation
Inherited States and Properties:	aria-atomic aria-busy (state) aria-controls aria-current (state) aria-describedat aria-describedby aria-disabled (state) aria-dropeffect aria-flowto aria-grabbed (state) aria-haspopup aria-hidden (state) aria-invalid (state) aria-label aria-labelledby aria-live aria-owns aria-relevant aria-valuemax aria-valuemin aria-valuenow aria-valuetext
Name From:	author
Accessible Name Required:	True
Children Presentational:	True
Implicit Value for Role:	Default for aria-orientation is horizontal.

spinbutton (role) §

A form of range that expects the user to select from among discrete choices.

A spinbutton typically allows the user to select from the given range through the use of an up and down button on the keyboard. Visibly, the current value is incremented or decremented until a maximum or minimum value is reached. Authors SHOULD ensure this functionality is accomplished programmatically through the use of up and down arrows on the keyboard.

Although a spinbutton is similar in appearance to many presentations of select, it is advisable to use spinbutton when working with known ranges (especially in the case of large ranges) as opposed to distinct options. For example, a spinbutton representing a range from 1 to 1,000,000 would provide much better performance than a select widget representing the same values.

Characteristics:

Characteristic	Value
Superclass Role:	input range
Required States and Properties:	aria-valuemax aria-valuemin aria-valuenow
Supported States and Properties:	aria-required
Inherited States and Properties:	aria-atomic aria-busy (state) aria-controls aria-current (state) aria-describedat aria-describedby aria-disabled (state) aria-dropeffect aria-flowto aria-grabbed (state) aria-haspopup aria-hidden (state) aria-invalid (state) aria-label aria-labelledby aria-live aria-owns aria-relevant aria-valuemax aria-valuemin aria-valuenow aria-valuetext
Name From:	author
Accessible Name Required:	True

---

status (role) §

A type of live region whose content is advisory information for the user but is not important enough to justify an alert, often but not necessarily presented as a status bar. See related alert.

Authors SHOULD ensure an element with role status does not receive focus as a result of change in status.

Status is a form of live region. If another part of the page controls what appears in the status, authors SHOULD make the relationship explicit with the aria-controls attribute.

Assistive technologies MAY reserve some cells of a Braille display to render the status.

> NOTE
>
> Elements with the role status have an implicit aria-live value of polite, and an implicit aria-atomic value of true.

Characteristics:

Characteristic	Value
Superclass Role:	section
Subclass Roles:	progressbar timer
Inherited States and Properties:	aria-atomic aria-busy (state) aria-controls aria-current (state) aria-describedat aria-describedby aria-disabled (state) aria-dropeffect aria-expanded (state) aria-flowto aria-grabbed (state) aria-haspopup aria-hidden (state) aria-invalid (state) aria-label aria-labelledby aria-live

	aria-owns aria-relevant
Name From:	author
Implicit Value for Role:	Default for aria-live is polite. Default for aria-atomic is true.

structure (abstract role) §

A document structural element.

Roles for document structure support the accessibility of dynamic web content by helping assistive technologies determine active content versus static document content. Structural roles by themselves do not all map to accessibility APIs, but are used to create widget roles or assist content adaptation for assistive technologies.

> NOTE
>
> structure is an abstract role used for the ontology. Authors should not use this role in content.

Characteristics:

Characteristic	Value
Is Abstract:	True
Superclass Role:	roletype
Subclass Roles:	document presentation rowgroup section sectionhead separator text
Inherited States and Properties:	aria-atomic aria-busy (state) aria-controls aria-current (state) aria-describedat aria-describedby aria-disabled (state) aria-dropeffect aria-flowto aria-grabbed (state) aria-haspopup aria-hidden (state) aria-invalid (state) aria-label aria-labelledby aria-live aria-owns aria-relevant
Name From:	n/a

switch (role) §

[ARIA 1.1] A type of checkbox that represents on/off values, as opposed to checked/unchecked values. See related checkbox.

The aria-checked attribute of a switch indicates whether the input is on (true) or off (false). The mixed value is not supported, and user agents MUST treat a mixed value as equivalent to false for this role.

> NOTE
>
> A switch provides approximately the same functionality as a checkbox and toggle button, but makes it possible for assistive technologies to present the widget in a fashion consistent with its on-screen appearance.

Characteristics:

Characteristic	Value
Superclass Role:	checkbox
Related Concepts:	button
Required States and Properties:	aria-checked
Inherited States and Properties:	aria-atomic aria-busy (state) aria-checked (state) aria-controls

	aria-current (state) aria-describedat aria-describedby aria-disabled (state) aria-dropeffect aria-flowto aria-grabbed (state) aria-haspopup aria-hidden (state) aria-invalid (state) aria-label aria-labelledby aria-live aria-owns aria-relevant
Name From:	contents author
Accessible Name Required:	True
Implicit Value for Role:	Default for aria-checked is false.

tab (role) §

A grouping label providing a mechanism for selecting the tab content that is to be rendered to the user.

If a tabpanel or item in a tabpanel has focus, the associated tab is the currently active tab in the tablist, as defined in Managing Focus. tablist elements, which contain a set of associated tab elements, are typically placed near a series of tabpanel elements, usually preceding it. See the WAI-ARIA Authoring Practices Guide [WAI-ARIA-PRACTICES] for details on implementing a tab set design pattern.

Authors MUST ensure elements with role tab are contained in, or owned by, an element with the role tablist.

Authors SHOULD ensure the tabpanel associated with the currently active tab is perceivable to the user.

For a single-selectable tablist, authors SHOULD hide other tabpanel elements from the user until the user selects the tab associated with that tabpanel. For a multi-selectable tablist, authors SHOULD ensure each visible tabpanel has its aria-expanded attribute set to true, and that the remaining hidden tabpanel elements have their aria-expanded attributes set to false.

In either case, authors SHOULD ensure that a selected tab has its aria-selected attribute set to true, that inactive tab elements have their aria-selected attribute set to false, and that the currently selected tab provides a visual indication that it is selected. In the absence of an aria-selected attribute on the current tab, user agents SHOULD indicate to assistive technologies through the platform accessibility API that the currently focused tab is selected.

Characteristics:

Characteristic	Value
Superclass Role:	sectionhead widget
Required Context Role:	tablist
Supported States and Properties:	aria-posinset aria-selected aria-setsize
Inherited States and Properties:	aria-atomic aria-busy (state) aria-controls aria-current (state) aria-describedat aria-describedby aria-disabled (state) aria-dropeffect aria-expanded (state) aria-flowto aria-grabbed (state) aria-haspopup aria-hidden (state) aria-invalid (state) aria-label aria-labelledby aria-live aria-owns aria-relevant
Name From:	contents author
Implicit Value for Role:	Default for aria-selected is false.

tablist (role) §

A list of tab elements, which are references to tabpanel elements.

To be keyboard accessible, authors SHOULD manage focus of descendants for all instances of this role, as described in

Managing Focus.

For a single-selectable tablist, authors SHOULD hide other tabpanel elements from the user until the user selects the tab associated with that tabpanel. For a multi-selectable tablist, authors SHOULD ensure each visible tabpanel has its aria-expanded attribute set to true, and that the remaining hidden tabpanel elements have their aria-expanded attributes set to false.

tablist elements are typically placed near usually preceding, a series of tabpanel elements. See the WAI-ARIA Authoring Practices Guide [WAI-ARIA-PRACTICES] for details on implementing a tab set design pattern.

NOTE

Elements with the role tablist have an implicit aria-orientation value of horizontal.

Characteristics:

Characteristic	Value
Superclass Role:	composite directory
Related Concepts:	DAISY Guide
Required Owned Elements:	tab
Supported States and Properties:	aria-level aria-multiselectable aria-orientation
Inherited States and Properties:	aria-activedescendant aria-atomic aria-busy (state) aria-controls aria-current (state) aria-describedat aria-describedby aria-disabled (state) aria-dropeffect aria-expanded (state) aria-flowto aria-grabbed (state) aria-haspopup aria-hidden (state) aria-invalid (state) aria-label aria-labelledby aria-live aria-owns aria-relevant
Name From:	author
Implicit Value for Role:	Default for aria-orientation is horizontal.

tabpanel (role) §

A container for the resources associated with a tab, where each tab is contained in a tablist.

Authors SHOULD associate a tabpanel element with its tab, either by using the aria-controls attribute on the tab to reference the tab panel, or by using the aria-labelledby attribute on the tab panel to reference the tab.

tablist elements are typically placed near, usually preceding, a series of tabpanel elements. See the WAI-ARIA Authoring Practices Guide [WAI-ARIA-PRACTICES] for details on implementing a tab set design pattern.

Characteristics:

Characteristic	Value
Superclass Role:	section
Inherited States and Properties:	aria-atomic aria-busy (state) aria-controls aria-current (state) aria-describedat aria-describedby aria-disabled (state) aria-dropeffect aria-expanded (state) aria-flowto aria-grabbed (state) aria-haspopup aria-hidden (state) aria-invalid (state) aria-label aria-labelledby aria-live aria-owns

	aria-relevant
Name From:	author
Accessible Name Required:	True

text (role) §

[ARIA 1.1] An element whose entire subtree should be exposed to accessibility APIs as plain text.

The text role allows authors to mark an element and all of its subtree contents as plain text.

Authors SHOULD NOT use the text role on interactive controls (buttons, links, etc.) or ancestors of those controls, because it could prevent users of assistive technologies from accessing the controls. Authors wishing to retain the semantics of subtree contents could consider the ARIA 1.0 presentation or ARIA 1.1 none role instead.

NOTE

As with any ARIA 1.1 role, authors may provide a fallback ARIA 1.0 role.

EXAMPLE 13

```
<p>I <span role="text img" aria-label="love">♥□</span> New York.</p>
<p>My <span role="text img" aria-label="heart">♥□</span> bleeds.</p>
<p><span role="text img" aria-label="3 of 5 stars">□□□□□□</span></p>
```

In a screen reader using an ARIA 1.1-capable browsers, the user would hear "I love New York." In an ARIA 1.0-capable browser, the user might hear "I, love image, New York."

NOTE

It is not necessary to use the fallback role if the implicit role for the element provides sufficient fallback. In this example, an ARIA 1.0-capable browser would not recognize the "text" role token, and fall back to the default image role.

EXAMPLE 14

```
<p>I <img src="icon.gif" alt="love" role="text"> New York.□</p>
<p>My <img src="icon.gif" alt="heart" role="text"> bleeds.□</p>
```

NOTE

The text role can also be used to flatten structural semantics without providing an explicit label. The following example would be presented to assistive technolgies as one static text element ("I like turtles") rather than three separate paragraphs.

EXAMPLE 15

```
<div role="text">
  <p>I</p>
  <p>like</p>
  <p>turtles</p>
</div>
```

Use caution when using the text role on structural elements. In particular, avoid using the text role on elements with interactive descendants.

EXAMPLE 16

```
<p role="text">
  <!-- Example of incorrect usage. -->
  This <a href="#">link</a> becomes presentational, so
  it is no longer a link. The paragraph is just plain text.
  This is probably not what the author intended.
</p>
```

Characteristics:

Characteristic	Value
Superclass Role:	structure
Inherited States and Properties:	aria-atomic aria-busy (state) aria-controls

	aria-current (state) aria-describedat aria-describedby aria-disabled (state) aria-dropeffect aria-flowto aria-grabbed (state) aria-haspopup aria-hidden (state) aria-invalid (state) aria-label aria-labelledby aria-live aria-owns aria-relevant
Name From:	contents author
Children Presentational:	True

textbox (role) §

A type of input that allows free-form text as its value.

If the aria-multiline attribute is true, the widget accepts line breaks within the input, as in an HTML textarea. Otherwise, this is a simple text box. The intended use is for languages that do not have a text input element, or cases in which an element with different semantics is repurposed as a text field.

> NOTE
>
> In most user agent implementations, the default behavior of the ENTER or RETURN key is different between the single-line and multi-line text fields in HTML. When user has focus in a single-line <input type="text"> element, the keystroke usually submits the form. When user has focus in a multi-line <textarea> element, the keystroke inserts a line break. The WAI-ARIA textbox role differentiates these types of boxes with the aria-multiline attribute, so authors are advised to be aware of this distinction when designing the field.

Characteristics:

Characteristic	Value
Superclass Role:	input
Subclass Roles:	searchbox
Related Concepts:	XForms input HTML textarea HTML input[type="text"]
Supported States and Properties:	aria-activedescendant aria-autocomplete aria-multiline aria-placeholder aria-readonly aria-required
Inherited States and Properties:	aria-atomic aria-busy (state) aria-controls aria-current (state) aria-describedat aria-describedby aria-disabled (state) aria-dropeffect aria-flowto aria-grabbed (state) aria-haspopup aria-hidden (state) aria-invalid (state) aria-label aria-labelledby aria-live aria-owns aria-relevant
Name From:	author
Accessible Name Required:	True

timer (role) §

A type of live region containing a numerical counter which indicates an amount of elapsed time from a start point, or the time remaining until an end point.

The text contents of the timer object indicate the current time measurement, and are updated as that amount changes. The

timer value is not necessarily machine parsable, but authors SHOULD update the text contents at fixed intervals, except when the timer is paused or reaches an end-point.

> NOTE
>
> Elements with the role timer have an implicit aria-live value of off.

Characteristics:

Characteristic	Value
Superclass Role:	status
Inherited States and Properties:	aria-atomic aria-busy (state) aria-controls aria-current (state) aria-describedat aria-describedby aria-disabled (state) aria-dropeffect aria-expanded (state) aria-flowto aria-grabbed (state) aria-haspopup aria-hidden (state) aria-invalid (state) aria-label aria-labelledby aria-live aria-owns aria-relevant
Name From:	author
Accessible Name Required:	True

---

toolbar (role) §

A collection of commonly used function buttons or controls represented in compact visual form.

The toolbar is often a subset of functions found in a menubar, designed to reduce user effort in using these functions. Authors MUST supply a label on each toolbar when the application contains more than one toolbar.

Authors MAY manage focus of descendants for all instances of this role, as described in Managing Focus.

> NOTE
>
> Elements with the role toolbar have an implicit aria-orientation value of horizontal.

Characteristics:

Characteristic	Value
Superclass Role:	group
Related Concepts:	menubar
Supported States and Properties:	aria-orientation
Inherited States and Properties:	aria-activedescendant aria-atomic aria-busy (state) aria-controls aria-current (state) aria-describedat aria-describedby aria-disabled (state) aria-dropeffect aria-expanded (state) aria-flowto aria-grabbed (state) aria-haspopup aria-hidden (state) aria-invalid (state) aria-label aria-labelledby aria-live aria-owns aria-relevant
Name From:	author
Implicit Value for Role:	Default for aria-orientation is horizontal.

---

tooltip (role) §

A contextual popup that displays a description for an element.

The tooltip typically becomes visible in response to a mouse hover, or after the owning element receives keyboard focus. In each of these cases, authors SHOULD display the tooltip after a short delay. The use of a WAI-ARIA tooltip is a supplement to the normal tooltip behavior of the user agent.

> NOTE
>
> Typical tooltip delays last from one to five seconds.

Authors SHOULD ensure that elements with the role tooltip are referenced through the use of aria-describedby before or at the time the tooltip is displayed.

Characteristics:

Characteristic	Value
Superclass Role:	section
Inherited States and Properties:	aria-atomic aria-busy (state) aria-controls aria-current (state) aria-describedat aria-describedby aria-disabled (state) aria-dropeffect aria-expanded (state) aria-flowto aria-grabbed (state) aria-haspopup aria-hidden (state) aria-invalid (state) aria-label aria-labelledby aria-live aria-owns aria-relevant
Name From:	contents author
Accessible Name Required:	True

---

tree (role) §

A type of list that may contain sub-level nested groups that can be collapsed and expanded.

To be keyboard accessible, authors SHOULD manage focus of descendants for all instances of this role, as described in Managing Focus.

> NOTE
>
> Elements with the role tree have an implicit aria-orientation value of vertical.

Characteristics:

Characteristic	Value
Superclass Role:	select
Subclass Roles:	treegrid
Required Owned Elements:	group → treeitem treeitem
Supported States and Properties:	aria-multiselectable aria-required
Inherited States and Properties:	aria-activedescendant aria-atomic aria-busy (state) aria-controls aria-current (state) aria-describedat aria-describedby aria-disabled (state) aria-dropeffect aria-expanded (state) aria-flowto aria-grabbed (state) aria-haspopup aria-hidden (state)

	aria-invalid (state) aria-label aria-labelledby aria-live aria-orientation aria-owns aria-relevant
Name From:	author
Accessible Name Required:	True
Implicit Value for Role:	Default for aria-orientation is vertical.

treegrid (role) §

A grid whose rows can be expanded and collapsed in the same manner as for a tree.

A treegrid is considered editable unless otherwise specified. To make a treegrid read-only, set the aria-readonly attribute of the treegrid to true. User Agents MUST implicitly propagate value of the treegrid element's aria-readonly attribute to all of its owned gridcell elements, and expose this to the accessibility API. An author MAY override an individual gridcell element's propagated aria-readonly attribute on the gridcell.

To be keyboard accessible, authors SHOULD manage focus of descendants for all instances of this role, as described in Managing Focus.

Characteristics:

Characteristic	Value
Superclass Role:	grid tree
Required Owned Elements:	row rowgroup → row
Inherited States and Properties:	aria-activedescendant aria-atomic aria-busy (state) aria-colcount aria-controls aria-current (state) aria-describedat aria-describedby aria-disabled (state) aria-dropeffect aria-expanded (state) aria-flowto aria-grabbed (state) aria-haspopup aria-hidden (state) aria-invalid (state) aria-label aria-labelledby aria-level aria-live aria-multiselectable aria-orientation aria-owns aria-readonly aria-relevant aria-required aria-rowcount
Name From:	author
Accessible Name Required:	True

treeitem (role) §

An option item of a tree. This is an element within a tree that may be expanded or collapsed if it contains a sub-level group of tree item elements.

A collection of treeitem elements to be expanded and collapsed are enclosed in an element with the group role.

Authors MUST ensure elements with role treeitem are contained in, or owned by, an element with the role group or tree.

Characteristics:

Characteristic	Value
Superclass Role:	listitem option
Required Context Role:	group tree
Inherited States and Properties:	aria-atomic aria-busy (state)

	aria-checked aria-controls aria-current (state) aria-describedat aria-describedby aria-disabled (state) aria-dropeffect aria-expanded (state) aria-flowto aria-grabbed (state) aria-haspopup aria-hidden (state) aria-invalid (state) aria-label aria-labelledby aria-level aria-live aria-owns aria-posinset aria-relevant aria-selected (state) aria-setsize
Name From:	contents author
Accessible Name Required:	True

widget (abstract role) §

An interactive component of a graphical user interface (GUI).

Widgets are discrete user interface objects with which the user can interact. Widget roles map to standard features in accessibility APIs. When the user navigates an element assigned any of the non-abstract subclass roles of widget, assistive technologies that typically intercept standard keyboard events SHOULD switch to an application browsing mode, and pass keyboard events through to the web application. The intent is to hint to certain assistive technologies to switch from normal browsing mode into a mode more appropriate for interacting with a web application; some user agents have a browse navigation mode where keys, such as up and down arrows, are used to browse the document, and this native behavior prevents the use of these keys by a web application.

> NOTE
>
> widget is an abstract role used for the ontology. Authors should not use this role in content.

Characteristics:

Characteristic	Value
Is Abstract:	True
Superclass Role:	roletype
Subclass Roles:	columnheader command composite gridcell input range row rowheader tab
Inherited States and Properties:	aria-atomic aria-busy (state) aria-controls aria-current (state) aria-describedat aria-describedby aria-disabled (state) aria-dropeffect aria-flowto aria-grabbed (state) aria-haspopup aria-hidden (state) aria-invalid (state) aria-label aria-labelledby aria-live aria-owns aria-relevant
Name From:	n/a

window (abstract role) §

A browser or application window.

Elements with this role have a window-like behavior in a graphical user interface (GUI) context, regardless of whether they are implemented as a native window in the operating system, or merely as a section of the document styled to look like a window.

> NOTE
>
> In the description of this role, the term "application" does not refer to the application role, which specifies specific assistive technology behaviors.

> NOTE
>
> window is an abstract role used for the ontology. Authors should not use this role in content.

Characteristics:

Characteristic	Value
Is Abstract:	True
Superclass Role:	roletype
Subclass Roles:	dialog
Supported States and Properties:	aria-expanded aria-modal
Inherited States and Properties:	aria-atomic aria-busy (state) aria-controls aria-current (state) aria-describedat aria-describedby aria-disabled (state) aria-dropeffect aria-flowto aria-grabbed (state) aria-haspopup aria-hidden (state) aria-invalid (state) aria-label aria-labelledby aria-live aria-owns aria-relevant
Name From:	author

6. Supported States and Properties §

6.1 Clarification of States versus Properties §

The terms "states" and "properties" refer to similar features. Both provide specific information about an object, and both form part of the definition of the nature of roles. In this document, states and properties are both treated as aria-prefixed markup attributes. However, they are maintained conceptually distinct to clarify subtle differences in their meaning. One major difference is that the values of properties (such as aria-labelledby) are often less likely to change throughout the application life-cycle than the values of states (such as aria-checked) which may change frequently due to user interaction. Note that the frequency of change difference is not a rule; a few properties, such as aria-valuetext are expected to change often. Because the distinction between states and properties is of little consequence to most web content authors, this specification refers to both "states" and "properties" simply as "attributes" whenever possible. See the definitions of state and property for more information.

6.2 Characteristics of States and Properties §

States and properties have the characteristics described in the following sections.

6.2.1 Related Concepts §

Advisory information about features from this or other languages that correspond to this state or property. While the correspondence may not be exact, it is useful to help understand the intent of the state or property.

6.2.2 Used in Roles §

Advisory information about roles that use this state or property. This information is provided to help understand the appropriate usage of the state or property. Use of a given state or property is not defined when used on roles other than those listed.

6.2.3 Inherits into Roles §

Advisory information about roles that inherit the state or property from an ancestor role.

6.2.4 Value §

Value type of the state or property. The value may be one of the following types:

true/false
: Value representing either true or false, with a default "false" value.

tristate
: Value representing true or false, with an intermediate "mixed" value. Default value is "false" unless otherwise specified.

true/false/undefined
: Value representing true or false, with a default "undefined" value indicating the state or property is not relevant.

ID reference
: Reference to the ID of another element in the same document

ID reference list
: A list of one or more ID references.

integer
: A numerical value without a fractional component.

number
: Any real numerical value.

string
: Unconstrained value type.

token
: One of a limited set of allowed values.

token list
: A list of one or more tokens.

URI
: A Uniform Resource Identifier as defined by RFC 3986 [RFC3986]. It may reference a separate document, or a content fragment identifier in a separate document, or a content fragment identifier within the same document.

The "undefined" value, when allowed on a state or property, is an explicit indication that the state or property is not set. The value is used on states and properties that support tokens, and the "undefined" value is a string that is one of the allowed tokens. It is also used on some states and properties that accept true/false values, when "undefined" has a different meaning than "false".

These are generic types for states and properties, but do not define specific representation. See State and Property Attribute Processing for details on how these values are expressed and handled in host languages.

6.3 Values for States and Properties §

Many state and properties accept a specific set of tokens as values. The allowed values and explanation of their meaning is shown after the table of characteristics. The default value, if defined, is shown in strong type, followed by the parenthetical term 'default'. When a value is indicated as the default, the user agent MUST follow the behavior prescribed by this value when the state or property is empty or undefined. Some roles also define what behavior to use when certain states or properties, that do not have default values, are not provided.

6.4 Global States and Properties §

Some state and properties are applicable to all host language elements regardless of whether a role is applied. The following global states and properties are supported by all roles and by all base markup elements.

- aria-atomic
- aria-busy (state)
- aria-controls
- aria-current (state)
- aria-describedat
- aria-describedby
- aria-disabled (state)
- aria-dropeffect
- aria-flowto
- aria-grabbed (state)
- aria-haspopup
- aria-hidden (state)
- aria-invalid (state)
- aria-label
- aria-labelledby
- aria-live
- aria-owns
- aria-relevant

Global states and properties are applied to the role roletype, which is the base role, and therefore inherit into all roles. To facilitate reading, they are not explicitly identified as either supported or inherited states and properties in the specification. Instead, the inheritance is indicated by a link to this section.

6.5 Taxonomy of WAI-ARIA States and Properties §

States and properties are categorized as follows:

1. Widget Attributes
2. Live Region Attributes
3. Drag-and-Drop Attributes

4. Relationship Attributes

6.5.1 Widget Attributes §

This section contains attributes specific to common user interface elements found on GUI systems or in rich internet applications which receive user input and process user actions. These attributes are used to support the widget roles.

- aria-autocomplete
- aria-checked
- aria-disabled
- aria-expanded
- aria-haspopup
- aria-hidden
- aria-invalid
- aria-label
- aria-level
- aria-modal
- aria-multiline
- aria-multiselectable
- aria-orientation
- aria-placeholder
- aria-pressed
- aria-readonly
- aria-required
- aria-selected
- aria-sort
- aria-valuemax
- aria-valuemin
- aria-valuenow
- aria-valuetext

Widget attributes might be mapped by a user agent to platform accessibility API state, for access by assistive technologies, or they might be accessed directly from the DOM. User agents MUST provide a way for assistive technologies to be notified when states change, either through DOM attribute change events or platform accessibility API events.

6.5.2 Live Region Attributes §

This section contains attributes specific to live regions in rich internet applications. These attributes may be applied to any element. The purpose of these attributes is to indicate that content changes may occur without the element having focus, and to provide assistive technologies with information on how to process those content updates. Some roles specify a default value value for the aria-live attribute specific to that role. An example of a live region is a ticker section that lists updating stock quotes.

- aria-atomic
- aria-busy
- aria-live
- aria-relevant

6.5.3 Drag-and-Drop Attributes §

This section lists attributes which indicate information about drag-and-drop interface elements, such as draggable elements and their drop targets. Drop target information will be rendered visually by the author and provided to assistive technologies through an alternate modality.

- aria-dropeffect
- aria-grabbed

For more information about using drag-and-drop, see Drag-and-Drop Support in the WAI-ARIA Authoring Practices ([WAI-ARIA-PRACTICES]).

6.5.4 Relationship Attributes §

This section lists attributes that indicate relationships or associations between elements which cannot be readily determined from the document structure.

- aria-activedescendant
- aria-colcount
- aria-colindex
- aria-colspan
- aria-controls
- aria-describedby
- aria-flowto
- aria-labelledby
- aria-owns
- aria-posinset
- aria-rowcount
- aria-rowindex
- aria-rowspan
- aria-setsize

6.6 Definitions of States and Properties (all aria-* attributes) §

Below is an alphabetical list of WAI-ARIA state and properties to be used by rich internet application authors. A detailed definition of each WAI-ARIA state and property follows this compact list.

aria-activedescendant
: Identifies the currently active descendant of a composite widget.

aria-atomic
: Indicates whether assistive technologies will present all, or only parts of, the changed region based on the change notifications defined by the aria-relevant attribute. See related aria-relevant.

aria-autocomplete
: Indicates whether user input completion suggestions are provided.

aria-busy
: Indicates whether an element, and its subtree, are currently being updated.

aria-checked
: Indicates the current "checked" state of checkboxes, radio buttons, and other widgets. See related aria-pressed and aria-selected.

aria-colcount
: [ARIA 1.1] Defines the total number of columns in a table, grid, or treegrid. See related aria-colindex.

aria-colindex
: [ARIA 1.1] Defines an element's column index or position with respect to the total number of columns within a table, grid, or treegrid. See related aria-colcount and aria-colspan.

aria-colspan
: [ARIA 1.1] Defines the number of columns spanned by a cell or gridcell within a table, grid, or treegrid. See related aria-colindex and aria-rowspan.

aria-controls
: Identifies the element (or elements) whose contents or presence are controlled by the current element. See related aria-owns.

aria-current
: [ARIA 1.1] Indicates the element that represents the current item within a container or set of related elements.

aria-describedat
: [ARIA 1.1] Specifies a URI referencing content that describes the object. See related aria-describedby.

aria-describedby
: Identifies the element (or elements) that describes the object. See related aria-labelledby.

aria-disabled
: Indicates that the element is perceivable but disabled, so it is not editable or otherwise operable. See related aria-hidden and aria-readonly.

aria-dropeffect
: Indicates what functions can be performed when the dragged object is released on the drop target. This allows assistive technologies to convey the possible drag options available to users, including whether a pop-up menu of choices is provided by the application. Typically, drop effect functions can only be provided once an object has been grabbed for a drag operation as the drop effect functions available are dependent on the object being dragged.

aria-expanded
: Indicates whether the element, or another grouping element it controls, is currently expanded or collapsed.

aria-flowto
: Identifies the next element (or elements) in an alternate reading order of content which, at the user's discretion, allows assistive technology to override the general default of reading in document source order.

aria-grabbed
: Indicates an element's "grabbed" state in a drag-and-drop operation.

aria-haspopup
: Indicates that the element has a popup context menu or sub-level menu.

aria-hidden
: Indicates whether the element is exposed to an accessibility API. See related aria-disabled.

aria-invalid
: Indicates the entered value does not conform to the format expected by the application.

aria-label
: Defines a string value value that labels the current element. See related aria-labelledby.

aria-labelledby
: Identifies the element (or elements) that labels the current element. See related aria-describedby.

aria-level
: Defines the hierarchical level of an element within a structure.

aria-live
: Indicates that an element will be updated, and describes the types of updates the user agents, assistive technologies, and user can expect from the live region.

aria-modal
: [ARIA 1.1] Indicates whether an element is modal when displayed.

aria-multiline
: Indicates whether a text box accepts multiple lines of input or only a single line.

aria-multiselectable
: Indicates that the user may select more than one item from the current selectable descendants.

aria-orientation
: Indicates whether the element and orientation is horizontal, vertical, or undefined.

aria-owns
: Identifies an element (or elements) in order to define a visual, functional, or contextual parent/child relationship between DOM elements where the DOM hierarchy cannot be used to represent the relationship. See related aria-controls.

aria-placeholder
: [ARIA 1.1] Represents a short hint (a word or short phrase) intended to aid the user with data entry when the control has no value. A hint could be a sample value or a brief description of the expected format.

aria-posinset
: Defines an element's number or position in the current set of listitems or treeitems. Not required if all elements in the set are present in the DOM. See related aria-setsize.

aria-pressed
: Indicates the current "pressed" state of toggle buttons. See related aria-checked and aria-selected.

aria-readonly

Indicates that the element is not editable, but is otherwise operable. See related aria-disabled.

aria-relevant
Indicates what notifications the user agent will trigger when the accessibility tree within a live region is modified. See related aria-atomic.

aria-required
Indicates that user input is required on the element before a form may be submitted.

aria-rowcount
[ARIA 1.1] Defines the total number of rows in a table, grid, or treegrid. See related aria-rowindex.

aria-rowindex
[ARIA 1.1] Defines an element's row index or position with respect to the total number of rows within a table, grid, or treegrid. See related aria-rowcount and aria-rowspan.

aria-rowspan
[ARIA 1.1] Defines the number of rows spanned by a cell or gridcell within a table, grid, or treegrid. See related aria-rowindex and aria-colspan.

aria-selected
Indicates the current "selected" state of various widgets. See related aria-checked and aria-pressed.

aria-setsize
Defines the number of items in the current set of listitems or treeitems. Not required if all elements in the set are present in the DOM. See related aria-posinset.

aria-sort
Indicates if items in a table or grid are sorted in ascending or descending order.

aria-valuemax
Defines the maximum allowed value for a range widget.

aria-valuemin
Defines the minimum allowed value for a range widget.

aria-valuenow
Defines the current value for a range widget. See related aria-valuetext.

aria-valuetext
Defines the human readable text alternative of aria-valuenow for a range widget.

---

aria-activedescendant (property) §

Identifies the currently active descendant of a composite widget.

This is used when a composite widget is responsible for managing its current active child to reduce the overhead of having all children be focusable. Examples include: multi-level lists, trees, and grids. In some implementations the user agent may use aria-activedescendant to tell assistive technologies that the active descendant has focus. Authors MAY use the aria-activedescendant attribute on the focused descendant of a composite widget; for example, on a textbox descendant of a combo box.

Authors SHOULD ensure that the element targeted by the aria-activedescendant attribute is either a descendant of the container in the DOM, or is a logical descendant as indicated by the aria-owns attribute. The user agent is not expected to validate that the active descendant is a descendant of the container. Authors SHOULD ensure that the currently active descendant is visible and in view (or scrolls into view) when focused.

Characteristics:

Characteristic	Value
Related Concepts:	SVG [SVG2] and DOM [DOM-Level-2-Core] active
Used in Roles:	composite group textbox
Inherits into Roles:	combobox grid listbox menu menubar radiogroup row searchbox select tablist toolbar tree treegrid
Value:	ID reference

---

aria-atomic (property) §

Indicates whether assistive technologies will present all, or only parts of, the changed region based on the change notifications defined by the aria-relevant attribute. See related aria-relevant.

Both accessibility APIs and the Document Object Model [DOM-Level-2-Core] provide events to allow the assistive technologies to determine changed areas of the document.

When the content of a live region changes, user agents SHOULD examine the changed element and traverse the ancestors to find the first element with aria-atomic set, and apply the appropriate behavior for the cases below.

1. If none of the ancestors have explicitly set aria-atomic, the default is that aria-atomic is false, and assistive technologies will only present the changed node to the user.

2. If aria-atomic is explicitly set to false, assistive technologies will stop searching up the ancestor chain and present only the changed node to the user.
3. If aria-atomic is explicitly set to true, assistive technologies will present the entire contents of the element, including the author-defined live region label if one exists.

When aria-atomic is true, assistive technologies MAY choose to combine several changes and present the entire changed region at once.

Characteristics:

Characteristic	Value
Used in Roles:	All elements of the base markup
Value:	true/false

Values:

Value	Description
false (default)	Assistive technologies will present only the changed node or nodes.
true	Assistive technologies will present the entire changed region as a whole, including the author-defined label if one exists.

---

aria-autocomplete (property) §

Indicates whether user input completion suggestions are provided.

For a textbox with the aria-autocomplete attribute set to either inline or both, authors SHOULD ensure that any auto-completed text is selected, so the user can type over it.

Characteristics:

Characteristic	Value
Related Concepts:	XForms selection attribute in select
Used in Roles:	combobox textbox
Inherits into Roles:	searchbox
Value:	token

Values:

Value	Description
both	A list of choices appears and the currently selected suggestion also appears inline.
inline	The system provides text after the caret as a suggestion for how to complete the field.
list	A list of choices appears from which the user can choose.
none (default)	No input completion suggestions are provided.

---

aria-busy (state) §

Indicates whether an element, and its subtree, are currently being updated.

The default is that aria-busy is false. If authors know that multiple parts of the same element need to be loaded or modified, they can set aria-busy to true when the first part is loaded, and then set aria-busy to false when the last part is loaded. When a widget is missing required owned elements due to script execution or loading, authors MUST mark a containing element with aria-busy equal to true. For example, until a page is fully initialized and complete, an author could mark the document element as busy. If there is an error updating the element, author MAY set the aria-invalid attribute to true.

Characteristics:

Characteristic	Value
Used in Roles:	All elements of the base markup
Value:	true/false

Values:

Value	Description
false (default):	There are no more expected updates for the element.
true	The element is still being updated.

---

aria-checked (state) §

Indicates the current "checked" state of checkboxes, radio buttons, and other widgets. See related aria-pressed and aria-selected.

The aria-checked attribute indicates whether the element is checked (true), unchecked (false), or represents a group of other elements that have a mixture of checked and unchecked values (mixed). Most inputs only support values of true and false,

but the mixed value is supported by certain tri-state inputs such as a checkbox or menuitemcheckbox.

The mixed value is not supported on radio, menuitemradio, switch or any element that inherits from these in the taxonomy, and user agents MUST treat a mixed value as equivalent to false for those roles.

Examples using the mixed value of tri-state inputs are covered in WAI-ARIA Authoring Practices [WAI-ARIA-PRACTICES]

Characteristics:

Characteristic	Value
Used in Roles:	checkbox option radio switch
Inherits into Roles:	menuitemcheckbox menuitemradio treeitem
Value:	tristate

Values:

Value	Description
false	The element supports being checked but is not currently checked.
mixed	Indicates a mixed mode value for a tri-state checkbox or menuitemcheckbox.
true	The element is checked.
undefined (default)	The element does not support being checked.

aria-colcount (property) §

[ARIA 1.1] Defines the total number of columns in a table, grid, or treegrid. See related aria-colindex.

If all of the columns are present in the DOM, it is not necessary to set this attribute as the user agent can automatically calculate the total number of columns. However, if only a portion of the columns is present in the DOM at a given moment, this attribute is needed to provide an explicit indication of the number of columns in the full table.

Authors MUST set the value of aria-colcount to an integer equal to the number of columns in the full table. If the total number of columns is unknown, authors MUST set the value of aria-colcount to -1 to indicate that the value should not be calculated by the user agent.

The following example shows a grid with 16 columns, of which columns 2, 3, 4, and 9 are displayed to the user.

EXAMPLE 17

```
<div role="grid" aria-colcount="16">
  <div role="rowgroup">
    <div role="row">
      <span role="columnheader" aria-colindex="2">First Name</span>
      <span role="columnheader" aria-colindex="3">Last Name</span>
      <span role="columnheader" aria-colindex="4">Company</span>
      <span role="columnheader" aria-colindex="9">Phone</span>
    </div>
  </div>
  <div role="rowgroup">
    <div role="row">
      <span role="gridcell" aria-colindex="2">Fred</span>
      <span role="gridcell" aria-colindex="3">Jackson</span>
      <span role="gridcell" aria-colindex="4">Acme, Inc.</span>
      <span role="gridcell" aria-colindex="9">555-1234</span>
    </div>
    <div role="row">
      <span role="gridcell" aria-colindex="2">Sara</span>
      <span role="gridcell" aria-colindex="3">James</span>
      <span role="gridcell" aria-colindex="4">Acme, Inc.</span>
      <span role="gridcell" aria-colindex="9">555-1235</span>
    </div>
    …
  </div>
</div>
```

Characteristics:

Characteristic	Value
Used in Roles:	grid
Inherits into Roles:	treegrid
Value:	integer

aria-colindex (property) §

[ARIA 1.1] Defines an element's column index or position with respect to the total number of columns within a table, grid, or treegrid. See related aria-colcount and aria-colspan.

If all of the columns are present in the DOM, it is not necessary to set this attribute as the user agent can automatically calculate the column index of each cell or gridcell. However, if only a portion of the columns is present in the DOM at a given moment, this attribute is needed to provide an explicit indication of the column of each cell or gridcell with respect to the full table.

Authors MUST set the value for aria-colindex to an integer greater than or equal to 1, greater than the aria-colindex value of any previous elements within the same row, and less than or equal to the number of columns in the full table. For a cell or gridcell which spans multiple columns, authors MUST set the value of aria-colindex to the start of the span.

If the set of columns which is present in the DOM is contiguous, and if there are no cells which span more than one row or column in that set, then authors MAY place aria-colindex on each row, setting the value to the index of the first column of the set. Otherwise, authors SHOULD place aria-colindex on all of the children or owned elements of each row.

The following example shows a grid with 16 columns, of which columns 2 through 5 are displayed to the user. Because the set of columns is contiguous, aria-colindex can be placed on each row.

EXAMPLE 18

```
<div role="grid" aria-colcount="16">
  <div role="rowgroup">
    <div role="row" aria-colindex="2">
      <span role="columnheader">First Name</span>
      <span role="columnheader">Last Name</span>
      <span role="columnheader">Company</span>
      <span role="columnheader">Address</span>
    </div>
  </div>
  <div role="rowgroup">
    <div role="row" aria-colindex="2">
      <span role="gridcell">Fred</span>
      <span role="gridcell">Jackson</span>
      <span role="gridcell">Acme, Inc.</span>
      <span role="gridcell">123 Broad St.</span>
    </div>
    <div role="row" aria-colindex="2">
      <span role="gridcell">Sara</span>
      <span role="gridcell">James</span>
      <span role="gridcell">Acme, Inc.</span>
      <span role="gridcell">123 Broad St.</span>
    </div>
   …
  </div>
</div>
```

The following example shows a grid with 16 columns, of which columns 2 through 5 are displayed to the user. While the set of columns is contiguous, some of the cells span multiple rows. As a result, aria-colindex needs to be placed on all of the owned elements of each row.

EXAMPLE 19

```
<div role="grid" aria-colcount="16">
  <div role="rowgroup">
    <div role="row">
      <span role="columnheader" aria-colindex="2">First Name</span>
      <span role="columnheader" aria-colindex="3">Last Name</span>
      <span role="columnheader" aria-colindex="4">Company</span>
      <span role="columnheader" aria-colindex="5">Address</span>
    </div>
  </div>
  <div role="rowgroup">
    <div role="row">
      <span role="gridcell" aria-colindex="2">Fred</span>
      <span role="gridcell" aria-colindex="3">Jackson</span>
      <span role="gridcell" aria-colindex="4" aria-rowspan="2">Acme, Inc.</span>
      <span role="gridcell" aria-colindex="5" aria-rowspan="2">123 Broad St.</span>
    </div>
    <div role="row">
      <span role="gridcell" aria-colindex="2">Sara</span>
      <span role="gridcell" aria-colindex="3">James</span>
    </div>
   …
  </div>
</div>
```

The following example shows a grid with 16 columns, of which columns 2, 3, 4, and 9 are displayed to the user. Because the set of columns is non-contiguous, aria-colindex needs to be placed on all of the owned elements of each row.

EXAMPLE 20

```
<div role="grid" aria-colcount="16">
  <div role="rowgroup">
    <div role="row">
      <span role="columnheader" aria-colindex="2">First Name</span>
      <span role="columnheader" aria-colindex="3">Last Name</span>
      <span role="columnheader" aria-colindex="4">Company</span>
      <span role="columnheader" aria-colindex="9">Phone</span>
    </div>
  </div>
```

```
  <div role="rowgroup">
    <div role="row">
      <span role="gridcell" aria-colindex="2">Fred</span>
      <span role="gridcell" aria-colindex="3">Jackson</span>
      <span role="gridcell" aria-colindex="4">Acme, Inc.</span>
      <span role="gridcell" aria-colindex="9">555-1234</span>
    </div>
    <div role="row">
      <span role="gridcell" aria-colindex="2">Sara</span>
      <span role="gridcell" aria-colindex="3">James</span>
      <span role="gridcell" aria-colindex="4">Acme, Inc.</span>
      <span role="gridcell" aria-colindex="9">555-1235</span>
    </div>
    …
  </div>
</div>
```

Characteristics:

Characteristic	Value
Used in Roles:	gridcell row
Inherits into Roles:	columnheader rowheader
Value:	integer

aria-colspan (property) §

[ARIA 1.1] Defines the number of columns spanned by a cell or gridcell within a table, grid, or treegrid. See related aria-colindex and aria-rowspan.

This attribute is intended for cells and gridcells which are not contained in a native table. When defining the column span of cells or gridcells in a native table, authors SHOULD use the host language's attribute instead of aria-colspan. If aria-colspan is used on an element for which the host language provides an equivalent attribute, user agents MUST ignore the value of aria-colspan and instead expose the value of the host language's attribute to assistive technologies.

Authors MUST set the value of aria-colspan to an integer greater than or equal to 1 and less than the value which would cause the cell or gridcell to overlap the next cell or gridcell in the same row.

Characteristics:

Characteristic	Value
Used in Roles:	gridcell
Inherits into Roles:	columnheader rowheader
Value:	integer

aria-controls (property) §

Identifies the element (or elements) whose contents or presence are controlled by the current element. See related aria-owns.

For example:

- A table of contents tree view may control the content of a neighboring document pane.
- A group of checkboxes may control what commodity prices are tracked live in a table or graph.
- A tab controls the display of its associated tab panel.

Characteristics:

Characteristic	Value
Used in Roles:	All elements of the base markup
Value:	ID reference list

aria-current (state) §

[ARIA 1.1] Indicates the element that represents the current item within a container or set of related elements.

The aria-current attribute is an enumerated type. Any value not included in the list of allowed values SHOULD be treated by user agents or assistive technologies as if the value true had been provided. If the attribute is not present or its value is an empty string, the default value of false applies and the aria-current state MUST NOT be exposed by user agents or assistive technologies.

The aria-current attribute is used when an element within a set of related elements is visually styled to indicate it is the current item in the set. For example:

- A page token used to indicate a link within a set of pagination links, where the link is visually styled to represent the currently-displayed page.
- A step token used to indicate a link within a step indicator for a step-based process, where the link is visually

styled to represent the current step.
- A location token used to indicate the image that is visually highlighted as the current component of a flow chart.
- A date token used to indicate the current date within a calendar.
- A time token used to indicate the current time within a timetable.

Authors SHOULD only mark one element in a set of elements as current with aria-current.

Authors SHOULD NOT use the aria-current attribute as a substitute for aria-selected in widgets where aria-selected has the same meaning. For example, in a tablist, aria-selected is used on a tab to indicate the currently-displayed tabpanel.

NOTE

In some use cases for widgets that support aria-selected, current and selected can have different meanings and can both be used within the same set of elements. For example, aria-current="page" can be used in a navigation tree to indicate which page is currently displayed, while aria-selected="true" indicates which page will be displayed if the user activates the treeitem. Furthermore, the same tree may support operating on one or more selected pages (treeitems) by way of a context menu containing options such as "delete" and "move."

Characteristics:

Characteristic	Value
Used in Roles:	All elements of the base markup
Value:	token

Values:

Value	Description
page	Represents the current page within a set of pages.
step	Represents the current step within a process.
location	Represents the current location within an environment or context.
date	Represents the current date within a collection of dates.
time	Represents the current time within a set of times.
true	Represents the current item within a set.
false (default)	Does not represent the current item within a set.

aria-describedat (property) §

[ARIA 1.1] Specifies a URI referencing content that describes the object. See related aria-describedby.

Authors MUST ensure the URI value of aria-describedat be a valid reference to a separate document, or a content fragment identifier in a separate document, or a content fragment identifier within the same document.

Authors SHOULD use native markup features and self-describing content where possible (e.g., accessible SVG charts, audio-described video, EPUB footnotes), and only link to external content for descriptions when no other mechanism is available in the host language.

User agents SHOULD provide a device-independent mechanism to allow a user to navigate the user agent to content referenced by the aria-describedat attribute. User agents SHOULD also provide a device-independent mechanism to return the user's focus from the descriptive content view to the original content view. For example, a user agent MAY provide access to the document or document fragment referenced by the aria-describedat attribute in a contextual menu associated with the object.

EDITOR'S NOTE

Editorial Note: JC 2014-12-10 (updated from 2014-12-05 and 2014-06-03). The RFC-2119 statements in the previous paragraph are subject to change. Several implementors have expressed concern that if the requirements remain as-is, aria-describedat would change the established ARIA pattern to not affect mainstream UI.

In order to provide input- and device-independent access to the associated descriptive content, authors SHOULD use the tabindex attribute to ensure the object is included in the default tab order.

Characteristics of aria-describedat

Characteristic	Value
Related Concepts:	epub:describedat in EPUB 3 longdesc in HTML 4[HTML401]
Used in Roles:	All elements of the base markup
Value:	URI

Example: Using aria-describedat to provide a detailed description for raster images displayed in SVG

This example is informative.

The following example demonstrates three potential ways to use aria-describedat to provide a detailed description of

raster images displayed as part of a larger SVG [SVG2] document. SVG was used primarily because there is no equivalent mechanism in the host language. The <desc> is arguably similar, but does not allow for structured sub-level content.

EXAMPLE 21

```
<svg xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink">

  <!-- aria-describedat linking to a separate document -->
  <image x="10" y="10" width="130" height="100"
    aria-label="A painting inspired by Alfred Tennyson's poem The Lady of Shalott"
    aria-describedat="http://example.com/shalott-description"
    xlink:href="http://upload.wikimedia.org/wikipedia/commons/8/83/JWW_TheLadyOfShallot_1888.jpg" />

  <!-- aria-describedat linking to a content fragment identifier in a separate document -->
  <image x="10" y="120" width="130" height="100"
    aria-label="A painting inspired by Alfred Tennyson's poem The Lady of Shalott"
    aria-describedat="http://example.com/descriptions#shalott"
    xlink:href="http://upload.wikimedia.org/wikipedia/commons/8/83/JWW_TheLadyOfShallot_1888.jpg" />

  <!-- aria-describedat linking to a content fragment identifier within the same document (similar to aria-describedby) -->
  <image x="10" y="230" width="130" height="100"
    aria-label="A painting inspired by Alfred Tennyson's poem The Lady of Shalott"
    aria-describedat="#shalott-description"
    xlink:href="http://upload.wikimedia.org/wikipedia/commons/8/83/JWW_TheLadyOfShallot_1888.jpg" />
  <g id="shalott-description">
    <text>
      <tspan role="heading" aria-level="1" x="10" y="360">The Lady of Shallot, oil on canvas by John William Waterhouse, 1888</tspan>
      <tspan x="10" y="380">A small boat on a stream, containing a woman with auburn hair who is surrounded by an ornate tapestry.</tspan>
      <tspan x="10" y="400">The prow of the boat is inscribed "The Lady of Shalott" and is adorned with candles, a crucifix, and a lanterr
      <tspan x="10" y="420">Her expression is one of ...</tspan>
    </text>
  </g>

</svg>
```

---

aria-describedby (property) §

Identifies the element (or elements) that describes the object. See related aria-labelledby.

The aria-labelledby attribute is similar to the aria-describedby in that both reference other elements to calculate a text alternative, but a label should be concise, where a description is intended to provide more verbose information.

The element or elements referenced by the aria-describedby comprise the entire description. Include ID references to multiple elements if necessary, or enclose a set of elements (e.g., paragraphs) with the element referenced by the ID.

Characteristics:

Characteristic	Value
Related Concepts:	Hint or Help in XForms [XFORMS10] Label in XForms Label in HTML [XHTML11] online help HTML table cell headers
Used in Roles:	All elements of the base markup
Value:	ID reference list

---

aria-disabled (state) §

Indicates that the element is perceivable but disabled, so it is not editable or otherwise operable. See related aria-hidden and aria-readonly.

For example, irrelevant options in a radio group may be disabled. Disabled elements might not receive focus from the tab order. For some disabled elements, applications might choose not to support navigation to descendants. In addition to setting the aria-disabled attribute, authors SHOULD change the appearance (grayed out, etc.) to indicate that the item has been disabled.

The state of being disabled applies to the current element and all focusable descendant elements of the element on which the aria-disabled attribute is applied.

Characteristics:

Characteristic	Value
Used in Roles:	All elements of the base markup
Value:	true/false

Values:

Value	Description
false (default)	The element is enabled.
true	The element and all focusable descendants are disabled and its value cannot be changed by the user.

aria-dropeffect (property) §

Indicates what functions can be performed when the dragged object is released on the drop target. This allows assistive technologies to convey the possible drag options available to users, including whether a pop-up menu of choices is provided by the application. Typically, drop effect functions can only be provided once an object has been grabbed for a drag operation as the drop effect functions available are dependent on the object being dragged.

More than one drop effect may be supported for a given element. Therefore, the value value of this attribute is a space-delimited set of tokens indicating the possible effects, or none if there is no supported operation. In addition to setting the aria-dropeffect attribute, authors SHOULD show a visual indication of potential drop targets.

Characteristics:

Characteristic	Value
Used in Roles:	All elements of the base markup
Value:	token list

Values:

Value	Description
copy	A duplicate of the source object will be dropped into the target.
execute	A function supported by the drop target is executed, using the drag source as an input.
link	A reference or shortcut to the dragged object will be created in the target object.
move	The source object will be removed from its current location and dropped into the target.
none (default)	No operation can be performed; effectively cancels the drag operation if an attempt is made to drop on this object. Ignored if combined with any other token value. e.g., 'none copy' is equivalent to a 'copy' value.
popup	There is a popup menu or dialog that allows the user to choose one of the drag operations (copy, move, link, execute) and any other drag functionality, such as cancel.

aria-expanded (state) §

Indicates whether the element, or another grouping element it controls, is currently expanded or collapsed.

For example, this indicates whether a portion of a tree is expanded or collapsed. In other instances, this may be applied to page sections to mark expandable and collapsible regions that are flexible for managing content density. Simplifying the user interface by collapsing sections may improve usability for all, including those with cognitive or developmental disabilities.

If the element with the aria-expanded attribute controls the expansion of another grouping container that is not 'owned by' the element, the author SHOULD reference the container by using the aria-controls attribute.

Characteristics:

Characteristic	Value
Related Concepts:	Tapered prompts in voice browsing. Switch in SMIL [SMIL].
Used in Roles:	button combobox document link section sectionhead separator window
Inherits into Roles:	alert alertdialog application article banner columnheader complementary contentinfo definition dialog directory form grid gridcell group heading img landmark list

	listbox listitem log main marquee math menu menubar navigation note progressbar radiogroup region row rowheader search select status tab tablist tabpanel timer toolbar tooltip tree treegrid treeitem
Value:	true/false/undefined

Values:

Value	Description
false	The element, or another grouping element it controls, is collapsed.
true	The element, or another grouping element it controls, is expanded.
undefined (default)	The element, or another grouping element it controls, is neither expandable nor collapsible; all its child elements are shown or there are no child elements.

aria-flowto (property) §

Identifies the next element (or elements) in an alternate reading order of content which, at the user's discretion, allows assistive technology to override the general default of reading in document source order.

When aria-flowto has a single IDREF, it allows assistive technologies to, at the user's request, forego normal document reading order and go to the targeted object. However, when aria-flowto is provided with multiple IDREFS, assistive technologies SHOULD present the referenced elements as path choices.

In the case of one or more IDREFS, user agents or assistive technologies SHOULD give the user the option of navigating to any of the targeted elements. The name of the path can be determined by the name of the target element of the aria-flowto attribute. Accessibility APIs can provide named path relationships.

Characteristics:

Characteristic	Value
Used in Roles:	All elements of the base markup
Value:	ID reference list

aria-grabbed (state) §

Indicates an element's "grabbed" state in a drag-and-drop operation.

When it is set to true it has been selected for dragging, false indicates that the element can be grabbed for a drag-and-drop operation, but is not currently grabbed, and undefined (or no value value) indicates the element cannot be grabbed (default).

When aria-grabbed is set to true, authors SHOULD update the aria-dropeffect attribute of all potential drop targets. When an element is not grabbed (the value is set to false, undefined, or the attribute is removed), authors SHOULD revert the aria-dropeffect attributes of the associated drop targets to none.

Characteristics:

Characteristic	Value
Used in Roles:	All elements of the base markup
Value:	true/false/undefined

Values:

Value	Description
false	Indicates that the element supports being dragged.

true	Indicates that the element has been "grabbed" for dragging.
undefined (default)	Indicates that the element does not support being dragged.

aria-haspopup (property) §

Indicates that the element has a popup context menu or sub-level menu.

This means that activation renders conditional content. Note that ordinary tooltips are not considered popups in this context.

A popup is generally presented visually as a group of items that appears to be on top of the main page content.

Characteristics:

Characteristic	Value
Related Concepts:	aria-controls User Agent Accessibility Guidelines [UAAG10] conditional content
Used in Roles:	All elements of the base markup
Value:	true/false

Values:

Value	Description
false (default)	The object has no popup.
true	Indicates the object has a popup, either as a descendant or referenced by aria-owns.

aria-hidden (state) §

Indicates whether the element is exposed to an accessibility API. See related aria-disabled.

User agents determine an element's hidden status based on whether it is rendered, and the rendering is usually controlled by CSS. For example, an element whose display property is set to none is not rendered. An element is considered hidden if it, or any of its ancestors are not rendered or have their aria-hidden attribute value set to true.

EXAMPLE 22

```
[aria-hidden="true"] { visibility: hidden; }
```

Authors MAY, with caution, use aria-hidden to hide visibly rendered content from assistive technologies only if the act of hiding this content is intended to improve the experience for users of assistive technologies by removing redundant or extraneous content. Authors using aria-hidden to hide visible content from screen readers MUST ensure that identical or equivalent meaning and functionality is exposed to assistive technologies.

NOTE

Authors are advised to use extreme caution and consider a wide range of disabilities when hiding visibly rendered content from assistive technologies. For example, a sighted, dexterity-impaired individual may use voice-controlled assistive technologies to access a visual interface. If an author hides visible link text "Go to checkout" and exposes similar, yet non-identical link text "Check out now" to the accessibility API, the user may be unable to access the interface they perceive using voice control. Similar problems may also arise for screen reader users. For example, a sighted telephone support technician may attempt to have the blind screen reader user click the "Go to checkout" link, which they may be unable to find using a type-ahead item search ("Go to…").

NOTE

At the time of this writing, aria-hidden="false" is known to work inconsistently in browsers. As future implementations improve, use caution and test thoroughly before relying on this approach.

NOTE

It is recommended that authors key visibility of elements off this attribute, rather than change visibility and separately update this property. CSS 2 provides a way to select on attribute values ([CSS2]). The following CSS declaration makes content visible unless the aria-hidden attribute is true; scripts need only update the value of this attribute to change visibility:

EXAMPLE 23

```
[aria-hidden="true"] { visibility: hidden; }
```

Characteristics:

Characteristic	Value
Used in Roles:	All elements of the base markup
Value:	true/false/undefined

Values:

Value	Description
false	The element is exposed to the accessibility API as if it was rendered.
true	The element is hidden from the accessibility API.
undefined (default)	The element's hidden state is determined by the user agent based on whether it is rendered.

aria-invalid (state) §

Indicates the entered value does not conform to the format expected by the application.

If the value is computed to be invalid or out-of-range, the application author SHOULD set this attribute to true. User agents SHOULD inform the user of the error. Application authors SHOULD provide suggestions for corrections if they are known.

When the user attempts to submit data involving a field for which aria-required is true, authors MAY use the aria-invalid attribute to signal there is an error. However, if the user has not attempted to submit the form, authors SHOULD NOT set the aria-invalid attribute on required widgets simply because the user has not yet entered data.

For future expansion, the aria-invalid attribute is an enumerated type. Any value not recognized in the list of allowed values MUST be treated by user agents as if the value true had been provided. If the attribute is not present, or its value is false, or its value is an empty string, the default value of false applies.

Characteristics:

Characteristic	Value
Related Concepts:	XForms [XFORMS10] 'invalid' event http://www.w3.org/TR/2006/REC-xforms-20060314/slice4.html#evt-revalidate. This state is true if a form field is required but empty. However, the XForms valid property would be set to false.
Used in Roles:	All elements of the base markup
Value:	token

Values:

Value	Description
grammar	A grammatical error was detected.
false (default)	There are no detected errors in the value.
spelling	A spelling error was detected.
true	The value entered by the user has failed validation.

aria-label (property) §

Defines a string value value that labels the current element. See related aria-labelledby.

The purpose of aria-label is the same as that of aria-labelledby. It provides the user with a recognizable name of the object. The most common accessibility API mapping for a label is the accessible name property.

If the label text is visible on screen, authors SHOULD use aria-labelledby and SHOULD NOT use aria-label. There may be instances where the name of an element cannot be determined programmatically from the content of the element, and there are cases where providing a visible label is not the desired user experience. Most host languages provide an attribute that could be used to name the element (e.g., the title attribute in HTML [HTML401]), yet this could present a browser tooltip. In the cases where a visible label or visible tooltip is undesirable, authors MAY set the accessible name of the element using aria-label. As required by the text alternative computation, user agents give precedence to aria-labelledby over aria-label when computing the accessible name property.

Characteristics:

Characteristic	Value
Related Concepts:	A related concept is title in HTML [XHTML11].
Used in Roles:	All elements of the base markup
Value:	string

aria-labelledby (property) §

Identifies the element (or elements) that labels the current element. See related aria-describedby.

The purpose of aria-labelledby is the same as that of aria-label. It provides the user with a recognizable name of the object. The most common accessibility API mapping for a label is the accessible name property.

If the label text is visible on screen, authors SHOULD use aria-label. Use aria-label only if the interface is such that it is not possible to have a visible label on the screen. As required by the text alternative computation, user agents give

precedence to aria-label when computing the accessible name property.

The aria-labelledby attribute is similar to aria-describedby in that both reference other elements to calculate a text alternative, but a label should be concise, where a description is intended to provide more verbose information.

> NOTE
>
> The expected spelling of this property in U.S. English is "labeledby." However, the accessibility API features to which this property is mapped have established the "labelledby" spelling. This property is spelled that way to match the convention and minimize the difficulty for developers.

Characteristics:

Characteristic	Value
Related Concepts:	A related concept is label in XForms [XFORMS10] and HTML [XHTML11].
Used in Roles:	All elements of the base markup
Value:	ID reference list

---

aria-level (property) §

Defines the hierarchical level of an element within a structure.

This can be applied inside trees to tree items, to headings inside a document, to nested grids, nested tablists and to other structural items that may appear inside a container or participate in an ownership hierarchy. The value value for aria-level is an integer greater than or equal to 1.

Levels increase with depth. If the DOM ancestry does not accurately represent the level, authors SHOULD explicitly define the aria-level attribute.

This attribute is applied to elements that act as leaf nodes within the orientation of the set, for example, on elements with role treeitem rather than elements with role group. This means that multiple elements in a set may have the same value for this attribute. Although it would be less repetitive to provide a single value on the container, restricting this to leaf nodes ensures that there is a single way for assistive technologies to use the attribute.

If the DOM ancestry accurately represents the level, the user agent can calculate the level of an item from the document structure. This attribute can be used to provide an explicit indication of the level when that is not possible to calculate from the document structure or the aria-owns attribute. User agent support for automatic calculation of level may vary; authors SHOULD test with user agents and assistive technologies to determine whether this attribute is needed. If the author intends for the user agent to calculate the level, the author SHOULD omit this attribute.

> NOTE
>
> In the case of a treegrid, aria-level is supported on elements with the role row, not elements with role gridcell. At first glance, this may seem inconsistent with the application of aria-level on treeitem elements, but it is consistent in that the row acts as the leaf node within the vertical orientation of the grid, whereas the gridcell is a leaf node within the horizontal orientation of each row. Level is not supported on sets of cells within rows, so the aria-level attribute is applied to the element with the role row.

Characteristics:

Characteristic	Value
Used in Roles:	grid heading listitem row tablist
Inherits into Roles:	treegrid treeitem
Value:	integer

---

aria-live (property) §

Indicates that an element will be updated, and describes the types of updates the user agents, assistive technologies, and user can expect from the live region.

The values of this attribute are expressed in degrees of importance. When regions are specified as polite, assistive technologies will notify users of updates but generally do not interrupt the current task, and updates take low priority. When regions are specified as assertive, assistive technologies will immediately notify the user, and could potentially clear the speech queue of previous updates. Please refer to Live Region Properties and How to Use Them ([WAI-ARIA-PRACTICES], Section 5.2.1).

Politeness levels are essentially an ordering mechanism for updates and serve as a strong suggestion to user agents or assistive technologies. The value may be overridden by user agents, assistive technologies, or the user. For example, if assistive technologies can determine that a change occurred in response to a key press or a mouse click, the assistive technologies may present that change immediately even if the value of the aria-live attribute states otherwise.

Since different users have different needs, it is up to the user to tweak his or her assistive technologies' response to a live region with a certain politeness level from the commonly defined baseline. Assistive technologies may choose to implement increasing and decreasing levels of granularity so that the user can exercise control over queues and interruptions.

When the property is not set on an object that needs to send updates, the politeness level is the value of the nearest ancestor that sets the aria-live attribute.

The aria-live attribute is the primary determination for the order of presentation of changes to live regions. Implementations will also consider the default level of politeness in a role when the aria-live attribute is not set in the ancestor chain (e.g., log changes are polite by default). Items which are assertive will be presented immediately, followed by polite items. User agents or assistive technologies MAY choose to clear queued changes when an assertive change occurs. (e.g., changes in an assertive region may remove all currently queued changes)

When live regions are marked as polite, assistive technologies SHOULD announce updates at the next graceful opportunity, such as at the end of speaking the current sentence or when the user pauses typing. When live regions are marked as assertive, assistive technologies SHOULD notify the user immediately. Because an interruption may disorient users or cause them to not complete their current task, authors SHOULD NOT use the assertive value unless the interruption is imperative.

Characteristics:

Characteristic	Value
Used in Roles:	All elements of the base markup
Value:	token

Values:

Value	Description
assertive	Indicates that updates to the region have the highest priority and should be presented the user immediately.
off (default)	Indicates that updates to the region should not be presented to the user unless the used is currently focused on that region.
polite	Indicates that updates to the region should be presented at the next graceful opportunity, such as at the end of speaking the current sentence or when the user pauses typing.

aria-modal (property) §

[ARIA 1.1] Indicates whether an element is modal when displayed.

The aria-modal attribute is used to indicate that the presence of a "modal" element precludes usage of other content on the page. For example, when a modal dialog is displayed, it is expected that the user's interaction is limited to the contents of the dialog, until the modal dialog loses focus or is no longer displayed.

When a modal element is displayed, assistive technologies SHOULD navigate to the element unless focus has explicitly been set elsewhere. Assistive technologies MAY limit navigation to the modal element's contents. If focus moves to an element outside the modal element, assistive technologies SHOULD NOT limit navigation to the modal element.

When a modal element is displayed, authors MUST ensure the interface can be controlled using only decendants of the modal element. In other words, if a modal dialog has a close button, the button should be a descendant of the dialog. When a modal element is displayed, authors SHOULD mark all other contents as inert (such as "inert subtrees" in HTML) if the ability to do so exists in the host language.

Characteristics:

Characteristic	Value
Used in Roles:	window
Inherits into Roles:	alertdialog dialog
Value:	true/false

Values:

Value	Description
false (default)	Element is not modal.
true	Element is modal.

aria-multiline (property) §

Indicates whether a text box accepts multiple lines of input or only a single line.

> NOTE
>
> In most user agent implementations, the default behavior of the ENTER or RETURN key is different between the single-line and multi-line text fields in HTML. When user has focus in a single-line <input type="text"> element, the keystroke usually submits the form. When user has focus in a multi-line <textarea> element, the keystroke inserts a line break. The WAI-ARIA aria-multiline attribute, so authors are advised to be aware of this distinction when designing the field.

Characteristics:

Characteristic	Value
Used in Roles:	textbox
Inherits into Roles:	searchbox
Value:	true/false

Values:

Value	Description
false (default)	This is a single-line text box.
true	This is a multi-line text box.

aria-multiselectable (property) §

Indicates that the user may select more than one item from the current selectable descendants.

Authors SHOULD ensure that selected descendants have the aria-selected attribute set to true, and selectable descendant have the aria-selected attribute set to false. Authors SHOULD NOT use the aria-selected attribute on descendants that are not selectable.

> NOTE
>
> Lists and trees are examples of roles that might allow users to select more than one item at a time.

Characteristics:

Characteristic	Value
Used in Roles:	grid listbox tablist tree
Inherits into Roles:	treegrid
Value:	true/false

Values:

Value	Description
false (default)	Only one item can be selected.
true	More than one item in the widget may be selected at a time.

aria-orientation (property) §

Indicates whether the element and orientation is horizontal, vertical, or undefined.

> NOTE
>
> In ARIA 1.1, the default value for aria-orientation changed from horizontal to undefined. Implicit defaults are defined on some roles (e.g., slider defaults to horizontal; scrollbar defaults to vertical) but remain undefined on roles where an expected default orientation is ambiguous (e.g., radiogroup).

Characteristics:

Characteristic	Value
Used in Roles:	scrollbar select separator slider tablist toolbar
Inherits into Roles:	combobox listbox menu menubar radiogroup tree treegrid
Value:	token

Values:

Value	Description
horizontal	The element is oriented horizontally.
undefined (default)	The element's orientation is undefined.
vertical	The element is oriented vertically.

aria-owns (property) §

Identifies an element (or elements) in order to define a visual, functional, or contextual parent/child relationship between DOM elements where the DOM hierarchy cannot be used to represent the relationship. See related aria-controls.

The value value of the aria-owns attribute is a space-separated list of IDREFS that reference one or more elements in the document by ID. The reason for adding aria-owns is to expose a parent/child contextual relationship to assistive technologies that is otherwise impossible to infer from the DOM.

Authors SHOULD NOT use aria-owns as a replacement for the DOM hierarchy. If the relationship is represented in the DOM, do not use aria-owns. Authors MUST ensure that an element's ID is not specified in more than one other element's aria-owns attribute at any time. In other words, an element can have only one explicit owner.

Characteristics:

Characteristic	Value
Used in Roles:	All elements of the base markup
Value:	ID reference list

aria-placeholder (property) §

[ARIA 1.1] Represents a short hint (a word or short phrase) intended to aid the user with data entry when the control has no value. A hint could be a sample value or a brief description of the expected format.

Authors SHOULD NOT use aria-placeholder instead of a label as their purposes are different: The label indicates what kind of information is expected. The placeholder text is a hint about the expected value. See related aria-labelledby and aria-label.

Authors SHOULD present this hint to the user by displaying the hint text at any time the control's value is the empty string. This includes cases where the control first receives focus, and when users remove a previously-entered value.

NOTE

As is the case with the related HTML placeholder attribute, use of placeholder text as a replacement for a displayed label can reduce the accessibility and usability of the control for a range of users including older users and users with cognitive, mobility, fine motor skill or vision impairments. While the hint given by the control's label is shown at all times, the short hint given in the placeholder attribute is only shown before the user enters a value. Furthermore, placeholder text may be mistaken for a pre-filled value, and as commonly implemented the default color of the placeholder text provides insufficient contrast and the lack of a separate visible label reduces the size of the hit region available for setting focus on the control.

The following example shows a searchbox in which the user has entered a value:

EXAMPLE 24

```
<span id="label">Birthday:</span>
<div role="searchbox" aria-labelledby="label" aria-placeholder="MM-DD-YYYY">03-14-1879</div>
```

The following example shows the same searchbox in which the user has not yet entered a value or has removed a previously-entered value:

EXAMPLE 25

```
<span id="label">Birthday:</span>
<div role="searchbox" aria-labelledby="label" aria-placeholder="MM-DD-YYYY">MM-DD-YYYY</div>
```

Characteristics:

Characteristic	Value
Related Concepts:	HTML placeholder
Used in Roles:	textbox
Inherits into Roles:	searchbox
Value:	string

aria-posinset (property) §

Defines an element's number or position in the current set of listitems or treeitems. Not required if all elements in the set are present in the DOM. See related aria-setsize.

If all items in a set are present in the document structure, it is not necessary to set this attribute, as the user agent can automatically calculate the set size and position for each item. However, if only a portion of the set is present in the document structure at a given moment, this property is needed to provide an explicit indication of an element's position.

The following example shows items 5 through 8 in a set of 16.

EXAMPLE 26

```
<h2 id="label_fruit"> Available Fruit </h2>
<ul role="listbox" aria-labelledby="label_fruit">
  <li role="option" aria-setsize="16" aria-posinset="5"> apples </li>
  <li role="option" aria-setsize="16" aria-posinset="6"> bananas </li>
  <li role="option" aria-setsize="16" aria-posinset="7"> cantaloupes </li>
  <li role="option" aria-setsize="16" aria-posinset="8"> dates </li>
</ul>
```

Authors MUST set the value value for aria-posinset to an integer greater than or equal to 1, and less than or equal to the size of the set. Authors SHOULD use aria-setsize.

Characteristics:

Characteristic	Value
Used in Roles:	listitem option radio tab
Inherits into Roles:	menuitemradio treeitem
Value:	integer

aria-pressed (state) §

Indicates the current "pressed" state of toggle buttons. See related aria-checked and aria-selected.

Toggle buttons require a full press-and-release cycle to change their value value. Activating it once changes the value to true, and activating it another time changes the value back to false. A value of mixed means that the values of more than one item controlled by the button do not all share the same value. Examples of mixed-state buttons are described in WAI-ARIA Authoring Practices [WAI-ARIA-PRACTICES]. If the attribute is not present, the button is not a toggle button.

The aria-pressed attribute is similar but not identical to the aria-checked attribute. Operating systems support pressed on buttons and checked on checkboxes.

Characteristics:

Characteristic	Value
Used in Roles:	button
Value:	tristate

Values:

Value	Description
false	The element supports being pressed but is not currently pressed.
mixed	Indicates a mixed mode value for a tri-state toggle button.
true	The element is pressed.
undefined (default)	The element does not support being pressed.

aria-readonly (property) §

Indicates that the element is not editable, but is otherwise operable. See related aria-disabled.

This means the user can read but not set the value of the widget. Readonly elements are relevant to the user, and application authors SHOULD NOT restrict navigation to the element or its focusable descendants. Other actions such as copying the value of the element are also supported. This is in contrast to disabled elements, to which applications might not allow user navigation to descendants.

Examples include:

- A form element which represents a constant.
- Row or column headers in a spreadsheet grid.
- The result of a calculation such as a shopping cart total.

Characteristics:

Characteristic	Value
Related Concepts:	XForms [XFORMS10] Readonly
Used in Roles:	grid gridcell

	textbox
Inherits into Roles:	columnheader rowheader searchbox treegrid
Value:	true/false

Values:

Value	Description
false (default)	The user can set the value of the element.
true	The user cannot change the value of the element.

aria-relevant (property) §

Indicates what notifications the user agent will trigger when the accessibility tree within a live region is modified. See related aria-atomic.

The attribute is represented as a space delimited list of the following values: additions, removals, text; or a single catch-all value all.

This is used to describe semantically meaningful changes, as opposed to merely presentational ones. For example, nodes that are removed from the top of a log are merely removed for purposes of creating room for other entries, and the removal of them does not have meaning. However, in the case of a buddy list, removal of a buddy name indicates that they are no longer online, and this is a meaningful event. In that case aria-relevant will be set to all. When the aria-relevant attribute is not provided, the default value, additions text, indicates that text modifications and node additions are relevant, but that node removals are irrelevant.

> NOTE
>
> aria-relevant values of removals or all are to be used sparingly. Assistive technologies only need to be informed of content removal when its removal represents an important change, such as a buddy leaving a chat room.

> NOTE
>
> Text removals should only be considered relevant if one of the specified values is 'removals' or 'all'. For example, for a text change from 'foo' to 'bar' in a live region with a default aria-relevant value, the text addition ('bar') would be spoken, but the text removal ('foo') would not.

aria-relevant is an optional attribute of live regions. This is a suggestion to assistive technologies, but assistive technologies are not required to present changes of all the relevant types.

When aria-relevant is not defined, an element's value is inherited from the nearest ancestor with a defined value. Although the value is a token list, inherited values are not additive; the value provided on a descendant element completely overrides any inherited value from an ancestor element.

When text changes are denoted as relevant, user agents MUST monitor any descendant node change that affects the text alternative computation of the live region as if the accessible name were determined from contents (nameFrom: contents). For example, a text change would be triggered if the HTML alt attribute of a contained image changed. However, no change would be triggered if there was a text change to a node outside the live region, even if that node was referenced (via aria-labelledby) by an element contained in the live region.

Characteristics:

Characteristic	Value
Used in Roles:	All elements of the base markup
Value:	token list

Values:

Value	Description
additions	Element nodes are added to the accessibility tree within the live region.
additions text	Equivalent to the combination of values, "additions text".
all	Equivalent to the combination of all values, "additions removals text".
removals	Text content, a text alternative, or an element node within the live region is removed from the accessibility tree.
text	Text content or a text alternative is added to any descendant in the accessibility tree of the live region.

aria-required (property) §

Indicates that user input is required on the element before a form may be submitted.

For example, if the user needs to fill in an address field, the author will need to set the field's aria-required attribute

to true.

> NOTE
>
> The fact that the element is required is often presented visually (such as a sign or symbol after the widget). Using the aria-required attribute allows the author to explicitly convey to assistive technologies that an element is required.

Unless an exactly equivalent native attribute is available, host languages SHOULD allow authors to use the aria-required attribute on host language form elements that require input or selection by the user.

Characteristics:

Characteristic	Value
Related Concepts:	HTML 5 required
Used in Roles:	combobox gridcell listbox radiogroup spinbutton textbox tree
Inherits into Roles:	columnheader rowheader searchbox treegrid
Value:	true/false

Values:

Value	Description
false (default)	User input is not necessary to submit the form.
true	Users need to provide input on an element before a form is submitted.

aria-rowcount (property) §

[ARIA 1.1] Defines the total number of rows in a table, grid, or treegrid. See related aria-rowindex.

If all of the rows are present in the DOM, it is not necessary to set this attribute as the user agent can automatically calculate the total number of rows. However, if only a portion of the rows is present in the DOM at a given moment, this attribute is needed to provide an explicit indication of the number of rows in the full table.

Authors MUST set the value of aria-rowcount to an integer equal to the number of rows in the full table. If the total number of rows is unknown, authors MUST set the value of aria-rowcount to -1 to indicate that the value should not be calculated by the user agent.

The following example shows a grid with 2000 rows, of which the first row and rows 100 through 102 are displayed to the user.

EXAMPLE 27

```
<div role="grid" aria-rowcount="2000">
  <div role="rowgroup">
    <div role="row" aria-rowindex="1">
      <span role="columnheader">First Name</span>
      <span role="columnheader">Last Name</span>
      <span role="columnheader">Company</span>
      <span role="columnheader">Phone</span>
    </div>
  </div>
  <div role="rowgroup">
    <div role="row" aria-rowindex="100">
      <span role="gridcell">Fred</span>
      <span role="gridcell">Jackson</span>
      <span role="gridcell">Acme, Inc.</span>
      <span role="gridcell">555-1234</span>
    </div>
    <div role="row" aria-rowindex="101">
      <span role="gridcell">Sara</span>
      <span role="gridcell">James</span>
      <span role="gridcell">Acme, Inc.</span>
      <span role="gridcell">555-1235</span>
    </div>
    <div role="row" aria-rowindex="102">
      <span role="gridcell">Taylor</span>
      <span role="gridcell">Johnson</span>
      <span role="gridcell">Acme, Inc.</span>
      <span role="gridcell">555-1236</span>
    </div>
  </div>
</div>
```

Characteristics:

Characteristic	Value
Used in Roles:	grid
Inherits into Roles:	treegrid
Value:	integer

aria-rowindex (property) §

[ARIA 1.1] Defines an element's row index or position with respect to the total number of rows within a table, grid, or treegrid. See related aria-rowcount and aria-rowspan.

If all of the rows are present in the DOM, it is not necessary to set this attribute as the user agent can automatically calculate the index of each row. However, if only a portion of the rows is present in the DOM at a given moment, this attribute is needed to provide an explicit indication of each row's position with respect to the full table.

Authors MUST set the value for aria-rowindex to an integer greater than or equal to 1, greater than the aria-rowindex value of any previous rows, and less than or equal to the number of rows in the full table. For a cell or gridcell which spans multiple rows, authors MUST set the value of aria-rowindex to the start of the span.

Authors SHOULD place aria-rowindex on each row. Authors MAY also place aria-rowindex on all of the children or owned elements of each row.

The following example shows a grid with 2000 rows, of which the first row and rows 100 through 102 are displayed to the user.

EXAMPLE 28

```
<div role="grid" aria-rowcount="2000">
  <div role="rowgroup">
    <div role="row" aria-rowindex="1">
      <span role="columnheader">First Name</span>
      <span role="columnheader">Last Name</span>
      <span role="columnheader">Company</span>
      <span role="columnheader">Phone</span>
    </div>
  </div>
  <div role="rowgroup">
    <div role="row" aria-rowindex="100">
      <span role="gridcell">Fred</span>
      <span role="gridcell">Jackson</span>
      <span role="gridcell">>Acme, Inc.</span>
      <span role="gridcell">555-1234</span>
    </div>
    <div role="row" aria-rowindex="101">
      <span role="gridcell">Sara</span>
      <span role="gridcell">James</span>
      <span role="gridcell">>Acme, Inc.</span>
      <span role="gridcell">555-1235</span>
    </div>
    <div role="row" aria-rowindex="102">
      <span role="gridcell">Taylor</span>
      <span role="gridcell">Johnson</span>
      <span role="gridcell">>Acme, Inc.</span>
      <span role="gridcell">555-1236</span>
    </div>
  </div>
</div>
```

The following example shows the grid from the previous example with aria-rowindex also placed on all of the owned elements of each row.

EXAMPLE 29

```
<div role="grid" aria-rowcount="2000">
  <div role="rowgroup">
    <div role="row" aria-rowindex="1">
      <span role="columnheader" aria-rowindex="1">First Name</span>
      <span role="columnheader" aria-rowindex="1">Last Name</span>
      <span role="columnheader" aria-rowindex="1">Company</span>
      <span role="columnheader" aria-rowindex="1">Phone</span>
    </div>
  </div>
  <div role="rowgroup">
    <div role="row" aria-rowindex="100">
      <span role="gridcell" aria-rowindex="100">Fred</span>
      <span role="gridcell" aria-rowindex="100">Jackson</span>
      <span role="gridcell" aria-rowindex="100">>Acme, Inc.</span>
      <span role="gridcell" aria-rowindex="100">555-1234</span>
    </div>
    <div role="row" aria-rowindex="101">
      <span role="gridcell" aria-rowindex="101">Sara</span>
      <span role="gridcell" aria-rowindex="101">James</span>
      <span role="gridcell" aria-rowindex="101">>Acme, Inc.</span>
      <span role="gridcell" aria-rowindex="101">555-1235</span>
    </div>
    <div role="row" aria-rowindex="102">
```

```
      <span role="gridcell" aria-rowindex="102">Taylor</span>
      <span role="gridcell" aria-rowindex="102">Johnson</span>
      <span role="gridcell" aria-rowindex="102">>Acme, Inc.</span>
      <span role="gridcell" aria-rowindex="102">555-1236</span>
    </div>
  </div>
</div>
```

Characteristics:

Characteristic	Value
Used in Roles:	gridcell row
Inherits into Roles:	columnheader rowheader
Value:	integer

aria-rowspan (property) §

[ARIA 1.1] Defines the number of rows spanned by a cell or gridcell within a table, grid, or treegrid. See related aria-rowindex and aria-colspan.

This attribute is intended for cells and gridcells which are not contained in a native table. When defining the row span of cells or gridcells in a native table, authors SHOULD use the host language's attribute instead of aria-rowspan. If aria-rowspan is used on an element for which the host language provides an equivalent attribute, user agents MUST ignore the value of aria-rowspan and instead expose the value of the host language's attribute to assistive technologies.

Authors MUST set the value of aria-rowspan to an integer greater than or equal to 0 and less than the value which would cause the cell or gridcell to overlap the next cell or gridcell in the same column. Setting the value to 0 indicates that the cell or gridcell is to span all the remaining rows in the row group.

Characteristics:

Characteristic	Value
Used in Roles:	gridcell
Inherits into Roles:	columnheader rowheader
Value:	integer

aria-selected (state) §

Indicates the current "selected" state of various widgets. See related aria-checked and aria-pressed.

This attribute is used with single-selection and multiple-selection widgets:

1. Single-selection containers where the currently focused item is not selected. The selection normally follows the focus, and is managed by the user agent.
2. Multiple-selection containers. Authors SHOULD ensure that any selectable descendant of a container in which the aria-multiselectable attribute is true specifies a value of either true or false for the aria-selected attribute.

Any explicit assignment of aria-selected takes precedence over the implicit selection based on focus. If no DOM element in the widget is explicitly marked as selected, assistive technologies MAY convey implicit selection which follows the keyboard focus of the managed focus widget. If any DOM element in the widget is explicitly marked as selected, the user agent MUST NOT convey implicit selection for the widget.

Characteristics:

Characteristic	Value
Used in Roles:	gridcell option row tab
Inherits into Roles:	columnheader rowheader treeitem
Value:	true/false/undefined

Values:

Value	Description
false	The selectable element is not selected.
true	The selectable element is selected.
undefined (default)	The element is not selectable.

aria-setsize (property) §

Defines the number of items in the current set of listitems or treeitems. Not required if all elements in the set are present in the DOM. See related aria-posinset.

This property is marked on the members of a set, not the container element that collects the members of the set. To orient the user by saying an element is "item X out of Y," the assistive technologies would use X equal to the aria-posinset attribute and Y equal to the aria-setsize attribute.

If all items in a set are present in the document structure, it is not necessary to set this property, as the user agent can automatically calculate the set size and position for each item. However, if only a portion of the set is present in the document structure at a given moment (in order to reduce document size), this property is needed to provide an explicit indication of set size.

The following example shows items 5 through 8 in a set of 16.

EXAMPLE 30

```
<h2 id="label_fruit"> Available Fruit </h2>
<ul role="listbox" aria-labelledby="label_fruit">
  <li role="option" aria-setsize="16" aria-posinset="5"> apples </li>
  <li role="option" aria-setsize="16" aria-posinset="6"> bananas </li>
  <li role="option" aria-setsize="16" aria-posinset="7"> cantaloupes </li>
  <li role="option" aria-setsize="16" aria-posinset="8"> dates </li>
</ul>
```

Authors MUST set the value value for aria-setsize to an integer equal to the size of the set. Authors SHOULD use aria-posinset.

Characteristics:

Characteristic	Value
Used in Roles:	listitem option radio tab
Inherits into Roles:	menuitemradio treeitem
Value:	integer

aria-sort (property) §

Indicates if items in a table or grid are sorted in ascending or descending order.

Authors SHOULD only apply this property to table headers or grid headers. If the property is not provided, there is no defined sort order. For each table or grid, authors SHOULD apply aria-sort to only one header at a time.

Characteristics:

Characteristic	Value
Used in Roles:	columnheader rowheader
Value:	token

Values:

Value	Description
ascending	Items are sorted in ascending order by this column.
descending	Items are sorted in descending order by this column.
none (default)	There is no defined sort applied to the column.
other	A sort algorithm other than ascending or descending has been applied.

aria-valuemax (property) §

Defines the maximum allowed value for a range widget.

A range widget may start with a given value, which can be increased until a maximum value, defined by this property, is reached.

If the aria-valuenow has a known maximum and minimum, the author SHOULD provide properties for aria-valuemax and aria-valuemin. Authors MUST ensure the value of aria-valuemax is greater than or equal to the value of aria-valuemin.

Characteristics:

Characteristic	Value
Related Concepts:	XForms [XFORMS10] range
Used in Roles:	range scrollbar slider spinbutton

Inherits into Roles:	progressbar
Value:	number

aria-valuemin (property) §

Defines the minimum allowed value for a range widget.

A range widget may start with a given value, which can be decreased until a minimum value, defined by this property, is reached.

Declaring the minimum and maximum values allows alternate devices to react to arrow keys, validate the current value, or simply let the user know the size of the range. If the aria-valuenow has a known maximum and minimum, the author SHOULD provide properties for aria-valuemax and aria-valuemin.

Authors MUST ensure the value of aria-valuemin is less than or equal to the value of aria-valuemax.

Characteristics:

Characteristic	Value
Related Concepts:	XForms [XFORMS10] range
Used in Roles:	range scrollbar slider spinbutton
Inherits into Roles:	progressbar
Value:	number

aria-valuenow (property) §

Defines the current value for a range widget. See related aria-valuetext.

This property is used, for example, on a range widget such as a slider or progress bar.

If the current value is not known (for example, an indeterminate progress bar), the author SHOULD NOT set the aria-valuenow attribute. If the aria-valuenow attribute is absent, no information is implied about the current value. If the aria-valuenow has a known maximum and minimum, the author SHOULD provide properties for aria-valuemax and aria-valuemin.

The value of aria-valuenow is a decimal number. If the range is a set of numeric values, then aria-valuenow is one of those values. For example, if the range is [0, 1], a valid aria-valuenow is 0.5. A value outside the range, such as -2.5 or 1.1, is invalid.

For progressbar elements and scrollbar elements, assistive technologies SHOULD render the value to users as a percent, calculated as a position on the range from aria-valuemin to aria-valuemax if both are defined, otherwise the actual value with a percent indicator. For elements with role slider and spinbutton, assistive technologies SHOULD render the actual value to users.

When the rendered value cannot be accurately represented as a number, authors SHOULD use the aria-valuetext attribute in conjunction with aria-valuenow to provide a user-friendly representation of the range's current value. For example, a slider may have rendered values of small, medium, and large. In this case, the values of aria-valuetext would be one of the strings: small, medium, or large.

> NOTE
>
> If aria-valuetext is specified, assistive technologies render that instead of the value of aria-valuenow.

Characteristics:

Characteristic	Value
Related Concepts:	XForms [XFORMS10] range, start
Used in Roles:	range scrollbar slider spinbutton
Inherits into Roles:	progressbar
Value:	number

aria-valuetext (property) §

Defines the human readable text alternative of aria-valuenow for a range widget.

This property is used, for example, on a range widget such as a slider or progress bar.

If the aria-valuetext attribute is set, authors SHOULD also set the aria-valuenow attribute, unless that value is unknown (for example, on an indeterminate progressbar).

Authors SHOULD only set the aria-valuetext attribute when the rendered value cannot be meaningfully represented as a number.

For example, a slider may have rendered values of small, medium, and large. In this case, the values of aria-valuenow could range from 1 through 3, which indicate the position of each value in the value space, but the aria-valuetext would be one of the strings: small, medium, or large. If the aria-valuetext attribute is absent, the assistive technologies will rely solely on the aria-valuenow attribute for the current value.

If aria-valuetext is specified, assistive technologies SHOULD render that value instead of the value of aria-valuenow.

Characteristics:

Characteristic	Value
Related Concepts:	XForms [XFORMS10] range, start
Used in Roles:	range
Inherits into Roles:	progressbar scrollbar slider spinbutton
Value:	string

7. Implementation in Host Languages §

The roles, state, and properties defined in this specification do not form a complete web language or format. They are intended to be used in the context of a host language. This section discusses how host languages are to implement WAI-ARIA, to ensure that the markup specified here will integrate smoothly and effectively with the host language markup.

Although markup languages look alike superficially, they do not share language definition infrastructure. To accommodate differences in language-building approaches, the requirements are both general and modularization-specific. While allowing for differences in how the specifications are written, the intent is to maintain consistency in how the WAI-ARIA information looks to authors and how it is manipulated in the DOM by scripts.

WAI-ARIA roles, states, and properties are implemented as attributes of elements. Roles are applied by placing their names among the tokens appearing in the value value of a host-language-provided role attribute. States and properties each get their own attribute, with values as defined for each particular state or property in this specification. The name of the attribute is the aria-prefixed name of the state or property.

7.1 Role Attribute §

An implementing host language will provide an attribute with the following characteristics:

- The attribute name MUST be role;
- The attribute value value MUST allow a token list as the value;
- The appearance of the name literal of any concrete WAI-ARIA role as one of these tokens MUST NOT in and of itself make the attribute value illegal in the host-language syntax; and
- The first name literal of a non-abstract WAI-ARIA role in the list of tokens in the role attribute defines the role according to which the user agent MUST process the element. User Agent processing for roles is defined in the WAI-ARIA User Agent Implementation Guide [WAI-ARIA-IMPLEMENTATION].

7.2 State and Property Attributes §

An implementing host language MUST allow attributes with the following characteristics:

- The attribute name is the name of any state or property identified in the Supported States and Properties section, such as aria-busy, aria-selected, aria-activedescendant, aria-valuetext;
- The syntax does NOT prevent the attribute from appearing anywhere that it is applicable, as specified in this specification;
- When these attributes appear in a document instance, the attributes will be processed as defined in this specification.

Host languages that support XML Namespaces [XML-NAMES] MAY require that WAI-ARIA attributes be used with a namespace. In this case, the namespace for WAI-ARIA state and property attributes MUST be http://www.w3.org/ns/wai-aria/. To use WAI-ARIA in host languages that do not explicitly describe support for it, authors SHOULD use this namespace as well, if the host language supports namespaces and there is expectation that user agents will recognize the WAI-ARIA namespace. The namespace prefix is not defined by this specification but generally is expected to be "aria".

NOTE

The WAI-ARIA state and property attributes have a naming convention such that they all begin with the string "aria-". This is not a namespace prefix, it is a part of the state or property name. Therefore, when using WAI-ARIA states and properties with namespace prefixes, the complete attribute name will be like "aria:aria-foo".

Some host languages do not use namespaces with WAI-ARIA state and property attributes, either because the host language does not support namespaces or because the designers wish to incorporate WAI-ARIA into the core feature set. In these host languages, the namespace name for these attributes has no value. The names of these attributes do not have a prefix offset by a colon; in the terms of namespaces they are unprefixed attribute names. The ECMAScript binding of the DOM interface getAttributeNS for example, treats an empty string ("") as representing this condition, so that both getAttribute("aria-busy") and getAttributeNS("", "aria-busy") access the same aria-busy attribute in the DOM.

NOTE

According to the requirements of this section, some user agents recognize WAI-ARIA state and property attributes with namespaces, some without namespaces, and some might recognize both. Authors are advised to be aware of which form is supported for the host language they are using. Unless the host language and supporting user agents explicitly indicate that the namespace is required, authors are advised to use the attribute without namespaces. Even user agents that support namespaces generally do not publish namespaced WAI-ARIA states and properties to accessibility APIs. In particular, current implementations of HTML, including XHTML, do not support this namespace.

7.3 Focus Navigation §

An implementing host language MUST provide support for the author to make all interactive elements focusable, that is, any renderable or event-receiving elements. An implementing host language MUST provide a facility to allow web authors to define whether these focusable, interactive elements appear in the default tab navigation order. The tabindex attribute in HTML 5 is an example of one implementation.

7.4 Implicit WAI-ARIA Semantics §

WAI-ARIA is designed to provide semantic information about objects when host languages lack native semantics for the object. WAI-ARIA is designed, however, to provide additional semantics for many host languages. Furthermore, host languages over time can evolve and provide new native features that correspond to WAI-ARIA features. Therefore, there are many situations in which WAI-ARIA semantics are redundant with host language semantics.

These host language features can be viewed as having "implicit WAI-ARIA semantics". User agent processing of features with implicit WAI-ARIA semantics would be similar to the processing for the WAI-ARIA feature. The processing might not be identical because of lexical differences between the host language feature and the WAI-ARIA feature, but generally the user agent would expose the same information to the accessibility API. Features with implicit WAI-ARIA semantics satisfy WAI-ARIA structural requirements such as required owned elements, required states and properties, etc. and do not require explicit WAI-ARIA semantics to be provided.

For example, if an element with the functionality already exists, such as a checkbox or radio button, use the native semantics of the host language. WAI-ARIA markup is only intended to be used to enhance the native semantics (e.g., indicating that the element is required with aria-required), or to change the semantics to a different purpose form the standard functionality of the element.

Implicit WAI-ARIA semantics affect the conflict resolution procedures in the following section, Conflicts with Host Language Semantics. Therefore, implicit WAI-ARIA semantics need to be defined in a normative specification, such as the host language specification or the WAI-ARIA User Agent Implementation Guide [WAI-ARIA-IMPLEMENTATION].

7.5 Conflicts with Host Language Semantics §

WAI-ARIA roles, states, and properties are intended to add semantic information when native host language elements with these semantics are not available, and are generally used on elements that have no native semantics of their own. They can also be used on elements that have similar but non-identical semantics (for example, a nested list could be used to represent a tree structure). This method can be part of a fallback strategy for older browsers that have no WAI-ARIA implementation, or because native presentation of the repurposed element reduces the amount of style and/or script needed. Except for the cases outlined below, user agents MUST always use the WAI-ARIA semantics to define how it exposes the element to accessibility APIs, rather than using the host language semantics.

In addition to these normal situations in which WAI-ARIA is expected to override native semantics, there are elements that are inappropriate to override with WAI-ARIA. This could be because identical host language semantics exist, so WAI-ARIA is not needed, or because semantics from WAI-ARIA directly conflict with host language semantics. When a feature in the host language with identical role semantics and values is available, and the author has no compelling reason to avoid using the host language feature, authors SHOULD use the host language features rather than repurpose other elements with WAI-ARIA.

Host languages can have features that have implicit WAI-ARIA semantics corresponding to roles. When a WAI-ARIA role is provided, user agents MUST use the semantic of the WAI-ARIA role for processing, not the native semantic, unless the role requires WAI-ARIA states and properties whose attributes are explicitly forbidden on the native element by the host language. Values for roles do not conflict in the same way as values for states and properties (for example, the HTML 'checked' attribute and the 'aria-checked' attribute could have conflicting values), and authors are expected to have valid reason to provide a WAI-ARIA role even on elements that would not normally be repurposed.

When WAI-ARIA states and properties correspond to host language features that have the same implicit WAI-ARIA semantic, it can be particularly problematic to use the WAI-ARIA feature. If the WAI-ARIA feature and the host language feature are both provided but their values are not kept in sync, user agents and assistive technologies cannot know which value to use. Therefore, to prevent providing conflicting states and properties to assistive technologies, host languages MUST explicitly declare where the use of WAI-ARIA attributes on each host language element conflicts with native attributes for that element. When a host language declares a WAI-ARIA attribute to be in direct semantic conflict with a native attribute for a given element, user agents MUST ignore the WAI-ARIA attribute and instead use the host language attribute with the same implicit semantic.

Host languages MAY document features that cannot be overridden with WAI-ARIA (these are called "strong native semantics"). These can be features that have implicit WAI-ARIA semantics, as well as features where the processing would be uncertain if the semantics were changed with WAI-ARIA. Conformance checkers MAY signal an error or warning when a WAI-ARIA role is used on elements with strong native semantics, but as described above, user agents MUST still use the value of the the semantic of the WAI-ARIA role when exposing the element to accessibility APIs.

7.6 State and Property Attribute Processing §

State and property attributes are included in host languages, and therefore syntax for representation of their value types is governed by the host language. For each of the value types defined in Value, an appropriate value type from the host language is used. Recommended correspondences between WAI-ARIA value types and various host language value types are listed in Mapping WAI-ARIA Value types to languages. This is a non-normative mapping in order to accommodate new host languages

supporting WAI-ARIA.

The list value types—ID reference list and token list—allow more than one value of the given type to be provided. The values are separated by delimiter characters recognized by the host language for list attributes, such as space characters, commas, etc. Some languages may require a specific, single delimiter, while others may allow various delimiters.

Global states and properties are supported on any element in the host language. However, authors MUST only use non-global states and properties on elements with a role supporting the state or property; either defined as an explicit WAI-ARIA role, or as defined by the host language semantic matching an appropriate WAI-ARIA role. When a role attribute is added to an element, the semantics and behavior of the element, including support for WAI-ARIA states and properties, are augmented or overridden by the role behavior. User agents MUST ignore non-global states and properties used on an element without a role supporting the state or property; either defined as an explicit WAI-ARIA role, or as defined by the host language semantic matching an appropriate WAI-ARIA role. For example, the aria-valuetext attribute may be used on a progressbar.

When WAI-ARIA roles are used, supported states and properties that are not present in the DOM are treated according to their default value, unless they are required. For token states and properties, an attribute value that is a zero-length string ("") also corresponds to the default value. Therefore, user agents SHOULD treat token state and property attributes with a value of "" the same as they treat an absent attribute. Normally this corresponds to the default value (usually "undefined"), but if it is a required attribute, they signal an error (because a null value is the same as failing to provide the required attribute).

A. Schemata §

This section is non-normative.

WAI-ARIA roles, states, and properties are available in a number of machine-readable formats to support validation of content using WAI-ARIA attributes. WAI-ARIA is not finalized, however, so these files are subject to change without notice. Todo: Remove disclaimers about not final at rec.

It is not appropriate to use these document types for live content. These are made available only for download, to support local use in development, evaluation, and validation tools. Using these versions directly from the W3C server could cause automatic blockage, preventing them from loading.

If it is necessary to use schemata in content, follow guidelines to avoid excessive DTD traffic. For instance, use caching proxies to avoid fetching the schema each time it is used, or ensure software uses a local cache, such as with XML catalogs.

A.1 Roles Implementation §

The taxonomy for WAI-ARIA expressed in RDF is available from http://www.w3.org/WAI/ARIA/schemata/aria-1.rdf.

A.2 WAI-ARIA Attributes Module §

This module declares the WAI-ARIA attributes as a module that can be included in a modularized DTD. A sample XHTML DTD using this module follows. Note the WAI-ARIA attributes are in no namespace, and the attribute name begins with "aria-" to reduce the likelihood of collision with existing attributes.

This module is available from http://www.w3.org/MarkUp/DTD/aria-attributes-1.mod.

A.3 XHTML plus WAI-ARIA DTD §

This DTD extends XHTML 1.1 and adds the WAI-ARIA state and property attributes to all its elements. In order to provide broader keyboard support and conform with the Focus Navigation section above, it also adds the tabindex attribute to a wider set of elements.

This is not a formal document type and may be obsoleted by future formal XHTML DTDs that support WAI-ARIA.

The XHTML 1.1 plus WAI-ARIA DTD is available from http://www.w3.org/WAI/ARIA/schemata/xhtml-aria-1.dtd.

Documents written using this XHTML Family markup language can be validated using the above DTD. If a document author wants to facilitate such validation, they can include the following declaration at the top of their document:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML+ARIA 1.0//EN"
  "http://www.w3.org/WAI/ARIA/schemata/xhtml-aria-1.dtd">
```

However, note that when this DOCTYPE is present in a document, most user agents treat the document as generic XML rather than HTML. This causes them to be unable to support named character entities defined by the DTD (e.g., ©). Therefore, authors need to avoid use of named entities outside of the predefined entities in XML ([XML11], Section 4.6).

To avoid the above problem, authors can omit the above DOCTYPE statement. This causes user agents to treat the document as generic HTML with named character entity support as well as built-in ARIA support. However, it causes user agents to enter "quirks" mode which affects CSS rendering, and causes conformance checkers to fail the document due to the added ARIA attributes.

To avoid the issues of named character entity support and quirks mode, authors can instead use the following generic DOCTYPE declaration for HTML:

```
<!DOCTYPE html>
```

However, this still does not guarantee that documents will be validated by conformance checkers.

A.4 SGML Open Catalog Entry for XHTML+ARIA §

This section contains the SGML Open Catalog-format definition [SGML-CATALOG] of the public identifiers for XHTML+ARIA 1.0.

```
-- ..................................................................... --
-- File catalog  ....................................................... --

--  XHTML+ARIA Catalog Data File

        Revision:  $Revision: 1.3 $

        See "Entity Management", SGML Open Technical Resolution 9401 for detailed
        information on supplying and using catalog data. This document is available
        from OASIS at URL:

                <http://www.oasis-open.org/html/tr9401.html>

--

-- ..................................................................... --
-- SGML declaration associated with XHTML  ............................. --

OVERRIDE YES

SGMLDECL "xml1.dcl"

-- ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::: --

-- XHTML+ARIA modules          ........................................... --


PUBLIC "-//W3C//DTD XHTML+ARIA 1.0//EN" "xhtml-aria-1.dtd"


PUBLIC "-//W3C//ENTITIES XHTML ARIA Attributes 1.0//EN" "aria-attributes-1.mod"

-- End of catalog data  ................................................ --
-- ..................................................................... --
```

A.5 WAI-ARIA Attributes XML Schema Module §

This module declares the WAI-ARIA attributes as an XML Schema module that can be included in a modularized schema. Note the WAI-ARIA attributes are in no namespace, and the attribute name begins with "aria-" to reduce the likelihood of collision with existing attributes.

This module is available from http://www.w3.org/MarkUp/SCHEMA/aria-attributes-1.xsd.

A.6 HTML 4.01 plus WAI-ARIA DTD §

This standalone DTD adds WAI-ARIA state and property attributes to all elements in HTML 4.01, as well as a `role` attribute. In order to provide broader keyboard support, it also adds the `tabindex` attribute to a wider set of elements.

The DTD is based on the HTML 4.01 Transitional DTD, and includes all entity references needed to make it a standalone file. This is not an official W3C DTD and should be considered a derivative work of HTML 4.01.

Documents written using this markup language can be validated using the above DTD. If a document author wants to facilitate such validation, they can include the following declaration at the top of their document:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML+ARIA 1.0//EN"
        "http://www.w3.org/WAI/ARIA/schemata/html4-aria-1.dtd">
```

However, note that when this DOCTYPE is present in a document, most user agents treat the document as generic XML rather than HTML. This causes them to be unable to support named character entities defined by the DTD (e.g., ©). Therefore, authors need to avoid use of named entities outside of the predefined entities in XML ([XML11], Section 4.6).

To avoid the above problem, authors can omit the above DOCTYPE statement. This causes user agents to treat the document as generic HTML with named character entity support as well as built-in ARIA support. However, it causes user agents to enter "quirks" mode which affects CSS rendering, and causes conformance checkers to fail the document due to the added ARIA attributes.

To avoid the issues of named character entity support and quirks mode, authors can instead use the following generic DOCTYPE declaration for HTML:

```
<!DOCTYPE html>
```

However, this still does not guarantee that documents will be validated by conformance checkers.

The HTML Working Group is incorporating WAI-ARIA into HTML 5. Official support for WAI-ARIA in HTML will be provided in that specification. This DTD is made available only as a bridging solution for applications requiring DTD validation but not using HTML 5.

This module is available from http://www.w3.org/WAI/ARIA/schemata/html4-aria-1.dtd.

B. Mapping WAI-ARIA Value types to languages §

This section is non-normative.

EDITOR'S NOTE

Editorial note: The HTML 5 column of the table below is expected to be moved to the HTML 5 specification and become normative for that specification. Comments about ARIA lexicial processing in HTML 5 should be taken to the HTML Working Group, referencing ISSUE-129.

EDITOR'S NOTE

Editorial note: The suggested mappings for true/false values in HTML 5 use Keyword and enumerated attributes with allowed values of "true" and "false", instead of using the HTML 5 boolean value type. @@ can't rely on attribute absence because of default value in true/false/undefined case.

The table below provides recommended mappings between WAI-ARIA state and property types and attribute types from HTML 5, XML Schema Datatypes [XMLSCHEMA-2], SVG, and SGML.

Languages not listed below might have appropriate value types defined in the language. If they do not, we recommend XML Schema Datatypes for general purpose XML languages. Documents using DTDs instead of schemas will not be able to validate automatically and require additional processing on WAI-ARIA attributes.

WAI-ARIA type	HTML 5	XML Schema
true/false	Keyword and enumerated attributes with allowed values of "true" and "false"	boolean
true/false/undefined	Keyword and enumerated attributes with allowed values of "true", "false", and "undefined"	NMTOKEN with an enumeration constraint allowing values of "true", "false", and "undefined"
tristate	Keyword and enumerated attributes with allowed values of "true", "false", and "mixed"	NMTOKEN with an enumeration constraint allowing values of "true", "false", and "mixed"
number	Real number	decimal
integer	Non-negative integer	integer
token	Keyword and enumerated attributes	NMTOKEN with an enumeration constraint allowing values listed in the state or property definition
token list	Space-separated tokens	NMTOKENS with an enumeration constraintallowing values listed in the state or property definition
ID reference	The value of a defined id attribute on another element	IDREF
ID reference list	The value of one or more defined id attributes on other element(s), represented as Space-separated tokens	IDREFS
string	No value constraints	string

C. Change Log §

C.1 Substantive changes since the last public working draft §

- 9-Dec-2014: Removed legacy author requirements from aria-hidden that were once relevant to DOM-based screen readers.
- 14-Jan-2015: Added searchbox role.
- 15-Jan-2015: Added switch role.
- 22-Jan-2015: Added aria-current attribute.
- 29-Jan-2015: Made region a type of landmark. Add requirement that authors MUST give a region a brief label that describes the purpose of the content it contains. Remove the accessible name property from the section role. Change the superclass role from region to section for the following roles: alert, grid, landmark, list, log, status, and tabpanel. Remove region as a superclass role of article, making document the only superclass role of article.
- 09-Apr-2015: Added aria-placeholder attribute.
- 23-Apr-2015: Added aria-colcount, aria-rowcount, aria-colindex, aria-rowindex, aria-colspan, and aria-rowspan.

C.2 Other substantive changes since the WAI-ARIA 1.0 Recommendation §

- 17-Sept-2013: Added non-default value (false) for aria-checked to checkable widget roles: checkbox, menuitemcheckbox, menuitemradio, and radio.
- 17-Sept-2013: Added first draft of aria-describedat after much group deliberation.
- 17-Sept-2013: Added URI value type.
- 24-Apr-2014: aria-orientation now defaults to undefined, and is allowed on more roles with implicit defaults defined per role.
- 19-May-2014: radio no longer inherits from option, just from checkbox. radio now adds aria-posinset and aria-setsize.
- 19-May-2014: Added aria-posinset and aria-setsize to tab.
- 27-May-2014: Added placeholder for none role.
- 03-Aug-2014: Moved aria-selected from "supported" to "required" attribute list for option role.
- 05-Aug-2014: Changed rowgroup to subclass structure instead of group.
- 10-Nov-2014: Added aria-modal attribute.
- 10-Nov-2014: Added text role.

D. WAI-ARIA Role, State, and Property Quick Reference §

This section is non-normative.

The following table provides a quick reference to the supported states and properties for all WAI-ARIA roles that may be used in markup.

In addition to the states and properties shown in the table, the following global states and properties are supported on all roles.

Placeholder for global states and properties

Placeholder for quick reference table

E. Acknowledgments §

This section is non-normative.

The following people contributed to the development of this document.

E.1 Participants active in the PFWG at the time of publication §

- Christy Blew (University of Illinois at Urbana-Champaign)
- David Bolter (Mozilla Foundation)
- Michael Cooper (W3C/MIT)
- James Craig (Apple Inc.)
- Joanmarie Diggs (Igalia)
- Fred Esch (IBM Corporation)
- Steve Faulkner (The Paciello Group)
- John Foliot (Invited Expert)
- Christopher Gallelo (Microsoft Corporation)
- Bryan Garaventa (SSB BART Group)
- Scott González (JQuery Foundation)
- Billy Gregory (The Paciello Group)
- Karl Groves (The Paciello Group)
- Jon Gunderson (University of Illinois at Urbana-Champaign)
- Birkir Gunnarsson (Deque Systems, Inc.)
- Markus Gylling (DAISY Consortium)
- Mona Heath (University of Illinois at Urbana-Champaign)
- Susann Keohane (IBM Corporation)
- Matthew King (IBM Corporation)
- Jason Kiss (Department of Internal Affairs, New Zealand Government)
- Dominic Mazzoni (Google, Inc.)
- Shane McCarron (Invited Expert, Aptest)
- Charles McCathieNevile (Yandex)
- Mary Jo Mueller (IBM Corporation)
- James Nurthen (Oracle Corporation)
- Janina Sajka (Invited Expert, The Linux Foundation)
- Joseph Scheuhammer (Invited Expert, Inclusive Design Research Centre, OCAD University)
- Stefan Schnabel (SAP AG)
- Richard Schwerdtfeger (IBM Corporation)
- Lisa Seeman (Invited Expert)
- Cynthia Shelly (Microsoft Corporation)
- Alexander Surkov (Mozilla Foundation)
- Léonie Watson (The Paciello Group)
- Jason White (Educational Testing Service)
- Marco Zehe (Mozilla Foundation)
- Gottfried Zimmermann (Invited Expert, Access Technologies Group)

E.2 Other ARIA contributors, commenters, and previously active PFWG participants §

- Shadi Abou-Zahra (W3C)
- Jim Allan (TSB)
- Jonny Axelsson (Opera Software)
- David Baron (Mozilla Foundation)
- Art Barstow (Nokia Corporation)
- Simon Bates
- 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)
- Donald Evans (AOL)
- Chris Fleizach (Apple Inc.)
- Kelly Ford (Microsoft Corporation)
- Geoff Freed (Invited Expert, NCAM)
- Kentarou Fukuda (IBM Corporation)

- Bryan Garaventa
- Guido Geloso
- Ali Ghassemi
- Becky Gibson (IBM)
- Alfred S. Gilman
- Andres Gonzalez (Adobe Systems Inc.)
- James Graham
- Georgios Grigoriadis (SAP AG)
- Jeff Grimes (Oracle)
- Loretta Guarino Reid (Google, Inc.)
- Katie Haritos-Shea (Invited Expert)
- Barbara Hartel
- James Hawkins (Google, Inc.)
- Benjamin Hawkes-Lewis
- Sean Hayes (Microsoft Corporation)
- Jan Heck
- Shawn Henry
- Tina Homboe
- 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
- George Kerscher (International Digital Publishing Forum)
- Jason Kiss (New Zealand Government)
- Todd Kloots
- Johannes Koch
- Sam Kuper
- Earl Johnson (Sun)
- Jael Kurz
- Rajesh Lal (Nokia Corporation)
- Diego La Monica (International Webmasters Association / HTML Writers Guild (IWA-HWG))
- Aaron Leventhal (IBM Corporation)
- Gez Lemon (International Webmasters Association / HTML Writers Guild (IWA-HWG))
- Alex Li (SAP)
- Chris Lilley
- Thomas Logan (HiSoftware Inc.)
- William Loughborough (Invited Expert)
- Linda Mao (Microsoft)
- David MacDonald (Invited Expert, CanAdapt Solutions Inc.)
- Carolyn MacLeod
- Anders Markussen (Opera Software)
- Matthew May (Adobe Systems Inc.)
- Krzysztof Maczyński
- 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 (W3C)
- Mario Sánchez Prada (Samsung Electronics Co., Ltd. and Gnome Foundation)
- Martin Schaus (SAP AG)
- Doug Schepers (W3C)
- Matthias Schmitt
- 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
- Wu Wei (W3C / RITT)
- Ryan Williams (Oracle)
- Tom Wlodkowski
- Sam White (Apple Inc.)

E.3 Enabling funders §

This publication has been funded in part with Federal funds from the U.S. Department of Education, National Institute on Disability, Independent Living, and Rehabilitation Research (NIDILRR) under contract number ED-OSE-10-C-0067. 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.

F. References §

F.1 Normative references §

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

[WAI-ARIA-10]
James Craig; Michael Cooper et al. Accessible Rich Internet Applications (WAI-ARIA) 1.0. W3C Recommendation. URL: http://www.w3.org/TR/wai-aria/

F.2 Informative references §

[ACCNAME-AAM]
Joseph Scheuhammer; Michael Cooper; Andi Snow-Weaver; Aaron Leventhal et al. Accessible Name and Description: Computation and API Mappings 1.1. W3C Working Draft. URL: http://www.w3.org/TR/accname-aam-1.1/

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

[ATK]
Gnome Accessibility Toolkit 2.10. URL: https://developer.gnome.org/atk/2.10/

[AXAPI]
The Mac OS X Accessibility Protocol Mac OS 10.10. URL: https://developer.apple.com/library/mac/documentation/Cocoa/Reference/ApplicationKit/Protocols/NSAccessibility_Protocol/i

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

[DOM-Level-2-Core]
Arnaud Le Hors; Philippe Le Hégaret; Lauren Wood; Gavin Nicol; Jonathan Robie; Mike Champion; Steven B Byrne et al. Document Object Model (DOM) Level 2 Core Specification. 13 November 2000. W3C Recommendation. URL: http://www.w3.org/TR/DOM-Level-2-Core/

[HTML401]
Dave Raggett; Arnaud Le Hors; Ian Jacobs. HTML 4.01 Specification. 24 December 1999. W3C Recommendation. URL: http://www.w3.org/TR/html401

[HTML5]
Ian Hickson; Robin Berjon; Steve Faulkner; Travis Leithead; Erika Doyle Navara; Edward O'Connor; Silvia Pfeiffer. HTML5. 28 October 2014. W3C Recommendation. URL: http://www.w3.org/TR/html5/

[IAccessible2]
IAccessible2. URL: http://www.linuxfoundation.org/collaborate/workgroups/accessibility/iaccessible2

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

[MathML]
Patrick D F Ion; Robert R Miner. Mathematical Markup Language (MathML) 1.01 Specification. 7 July 1999. W3C Recommendation. URL: http://www.w3.org/TR/MathML/

[OWL-FEATURES]
Deborah McGuinness; Frank van Harmelen. OWL Web Ontology Language Overview. 10 February 2004. W3C Recommendation. URL: http://www.w3.org/TR/owl-features/

[RDF-CONCEPTS]
Graham Klyne; Jeremy Carroll. Resource Description Framework (RDF): Concepts and Abstract Syntax. 10 February 2004. W3C Recommendation. URL: http://www.w3.org/TR/rdf-concepts/

[RDF-SCHEMA]
Dan Brickley; Ramanathan Guha. RDF Schema 1.1. 25 February 2014. W3C Recommendation. URL: http://www.w3.org/TR/rdf-schema/

[RFC3986]
T. Berners-Lee; R. Fielding; L. Masinter. Uniform Resource Identifier (URI): Generic Syntax. January 2005. Internet Standard. URL: https://tools.ietf.org/html/rfc3986

[ROLE-ATTRIBUTE]
Shane McCarron et al. Role Attribute 1.0. 28 March 2013. W3C Recommendation. URL: http://www.w3.org/TR/role-attribute/

[SGML-CATALOG]
Paul Grosso. Entity Management: OASIS Technical Resolution 9401:1997 (Amendment 2 to TR 9401) 10 september 1007. Entity Management Subcommittee, SGML Open. URL: https://www.oasis-open.org/html/a401.htm

[SMIL]
Philipp Hoschka. Synchronized Multimedia Integration Language (SMIL 2.0) - [Second Edition]. 7 January 2005. W3C Recommendation. URL: http://www.w3.org/TR/SMIL/

[SVG2]

Nikos Andronikos; Tavmjong Bah; Amelia Bellamy-Royds; Brian Birtles; Cyril Concolato; Erik Dahlström; Chris Lilley; Cameron McCormack; Doug Schepers; Dirk Schulze; Richard Schwerdtfeger; Satoru Takagi; Jonathan Watt. Scalable Vector Graphics (SVG) 2. 9 April 2015. W3C Working Draft. URL: http://www.w3.org/TR/SVG2/

[UAAG10]
Ian Jacobs; Jon Gunderson; Eric Hansen. User Agent Accessibility Guidelines 1.0. 17 December 2002. W3C Recommendation. URL: http://www.w3.org/TR/UAAG10/

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

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

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

[WAI-ARIA-PRACTICES]
Joseph Scheuhammer; Michael Cooper. WAI-ARIA 1.0 Authoring Practices. 7 March 2013. W3C Working Draft. URL: http://www.w3.org/TR/wai-aria-practices/

[WAI-ARIA-PRIMER]
Lisa Pappas; Richard Schwerdtfeger; Michael Cooper. WAI-ARIA 1.0 Primer. 16 September 2010. W3C Working Draft. URL: http://www.w3.org/TR/wai-aria-primer/

[WAI-ARIA-ROADMAP]
Richard Schwerdtfeger. Roadmap for Accessible Rich Internet Applications (WAI-ARIA Roadmap). 4 February 2008. W3C Working Draft. URL: http://www.w3.org/TR/wai-aria-roadmap/

[WCAG20]
Ben Caldwell; Michael Cooper; Loretta Guarino Reid; Gregg Vanderheiden et al. Web Content Accessibility Guidelines (WCAG) 2.0. 11 December 2008. W3C Recommendation. URL: http://www.w3.org/TR/WCAG20/

[XFORMS10]
John Boyer. XForms 1.0 (Third Edition). 29 October 2007. W3C Recommendation. URL: http://www.w3.org/TR/xforms/

[XHTML11]
Shane McCarron; Masayasu Ishikawa. XHTML™ 1.1 - Module-based XHTML - Second Edition. 23 November 2010. W3C Recommendation. URL: http://www.w3.org/TR/xhtml11/

[XML-NAMES]
Tim Bray; Dave Hollander; Andrew Layman; Richard Tobin; Henry Thompson et al. Namespaces in XML 1.0 (Third Edition). 8 December 2009. W3C Recommendation. URL: http://www.w3.org/TR/xml-names

[XML11]
Tim Bray; Jean Paoli; Michael Sperberg-McQueen; Eve Maler; François Yergeau; John Cowan et al. Extensible Markup Language (XML) 1.1 (Second Edition). 16 August 2006. W3C Recommendation. URL: http://www.w3.org/TR/xml11/

[XMLSCHEMA-2]
Paul V. Biron; Ashok Malhotra. XML Schema Part 2: Datatypes Second Edition. 28 October 2004. W3C Recommendation. URL: http://www.w3.org/TR/xmlschema-2/