Accessible Rich Internet Applications (WAI-ARIA) Version 1.0

W3C Working Draft 6 August 2008

This version:
http://www.w3.org/TR/2008/WD-wai-aria-20080806/
Latest version:
http://www.w3.org/TR/wai-aria/
Previous version:
http://www.w3.org/TR/2008/WD-wai-aria-20080204/
Editors:
Michael Cooper, W3C
Rich Schwerdtfeger, IBM
Lisa Seeman, UB Access
Lisa Pappas, Society for Technical Communication

Abstract

Accessibility of Web content to people with disabilities requires semantic information about widgets, structures, and behaviors, in order to allow Assistive Technologies to make appropriate transformations. This specification provides an ontology of roles, states, and properties that set out an abstract model for accessible interfaces and can be used to improve the accessibility and interoperability of Web Content and Applications. This information can be mapped to accessibility frameworks that use this information to provide alternative access solutions. Similarly, this information can be used to change the rendering of content dynamically using different style sheet properties. The result is an interoperable method for associating behaviors with document-level markup. This document is part of the WAI-ARIA suite described in the 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/.

the Web Accessibility Initiative. The main innovation in this draft is the proposed approach to implement WAI-ARIA in host languages (see Implementation in Host Languages and Quality Assurance). The host language implementation requirements balance architectural goals with the constraints of present-day user agents. It seeks to address forward compatibility with *HTML 5* [*HTML5*], which is expected to be complete after WAI-ARIA.

Because of the above focus, this version of WAI-ARIA is published without corresponding updates to its companion documents: the *WAI-ARIA Primer* [*ARIA-PRIMER*], and *WAI-ARIA Best Practices* [*ARIA-PRACTICES*]. This version includes minor changes to the WAI-ARIA taxonomy since the previous version. Refer to the history of changes to WAI-ARIA for details.

Feedback on the model set out here is important to the success of the Web community in creating accessible Rich Internet Applications. The PFWG would like to know, particularly for this draft:

- Is the proposed method for adding WAI-ARIA to host languages effective and appropriate?
- Are conformance requirements clear, and sufficient but not excessive?

Additional questions:

- Does the role ontology provide the information and operations that people with disabilities need in order to access and operate richly interactive Web applications?
- Is the usage of roles, states, and properties clear?
- Is it clear how to create Rich Internet Applications with this added capability?

When addressing these questions, please consider them in the context of the companion documents. Comments on this document may be sent to public-pfwg-comments@w3.org (Archive). Comments requiring discussion should be copied to wai-xtech@w3.org (Archive). Comments should be made by **3 September 2008**.

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.

The disclosure obligations of the Participants of this group are described in the charter.

Table of Contents

1. Introduction

This section is *informative*.

The domain of Web accessibility defines how to make Web content usable by people with disabilities. People with some types of disabilities use Assistive Technology (AT) to interact with content. AT can transform the presentation of content into a format more suitable to the user, and can allow the user to interact in different ways than the author designed. In order to accomplish this, AT must understand the semantics of the content. Semantics are knowledge of roles, states, and properties, as a person would understand them, that apply to elements within the content. For instance, if a paragraph is semantically identified as such, AT can interact with it as a unit separable from the rest of the content, knowing the exact boundaries of that paragraph. A slider or tree widget is a more complex example, in which various parts of a widget each have semantics that must be properly identified for the computer to support effective interaction.

Established content technologies define semantics for elements commonly used in those technologies. However, new technologies can overlook some of the semantics required for accessibility. Furthermore, new authoring practices evolve which override the intended semantics—elements that have one defined semantic meaning in the technology are used with a different semantic meaning intended to be understood by the user.

For example, Rich Internet Applications developers can create a tree widget in HTML using CSS and JavaScript even though HTML lacks a semantic element for that. A different element must be used, possibly a list element with display instructions to make it look and behave like a tree widget. Assistive technology, however, must present the element in a different modality and the display instructions may not be applicable. The AT will present it as a list, which has very different display and interaction from a tree widget, and the user may be unsuccessful at understanding and operating the widget.

The incorporation of WAI-ARIA is a way for an author to provide proper type semantics on custom widgets (elements with repurposed semantics) to make these widgets accessible, usable and interoperable with assistive technologies. This specification identifies the types of widgets and structures that are 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 technology. Roles are a common property of platform Accessibility APIs which applications use to support assistive technologies. Assistive technology can then use the role information to provide effective presentation and interaction with these elements.

This role taxonomy currently includes interaction widget (user interface widget) and structural document (content organization) types of objects. The role taxonomy describes inheritance (widgets that are types of other widgets) and details what states and properties each role supports. When possible, information is provided about mapping of roles to accessibility APIs.

Roles are element types and should not change with time or user actions. Changing the role on an element from its inital value will be treated, via accessibility API events, as the removal of the old element and insertion of a new element with the new role.

Changeable states and properties of elements are also defined in this specification. States and Properties are used to declare important properties of an element that affect and describe interaction. These properties enable the user agent or operating system to properly handle the element even when these properties are altered dynamically by scripts. For example, alternative input and output technology such as screen readers, speech dictation software and on-screen keyboards must recognize the state of an element (such as: if an object is disabled, checked, focused, collapsed, hidden, etc.).

While it is possible for assistive technologies to access these properties through the *Document Object Model* [*DOM*], the preferred mechanism is for the user agent to map the States and Properties to the accessibility API of the operating system.

Figure 1.0 illustrates a typical *Document Object Model (DOM)* [*DOM*] node. Placed within the DOM node and the assistive technology is a box containing the contract provided by the user agent to the assistive technology. This data includes typical accessibility information found in the accessibility API for many of our accessible platforms for GUIs (role, state, caret, selection, event notification, parent/child information, relationship, and descriptions).

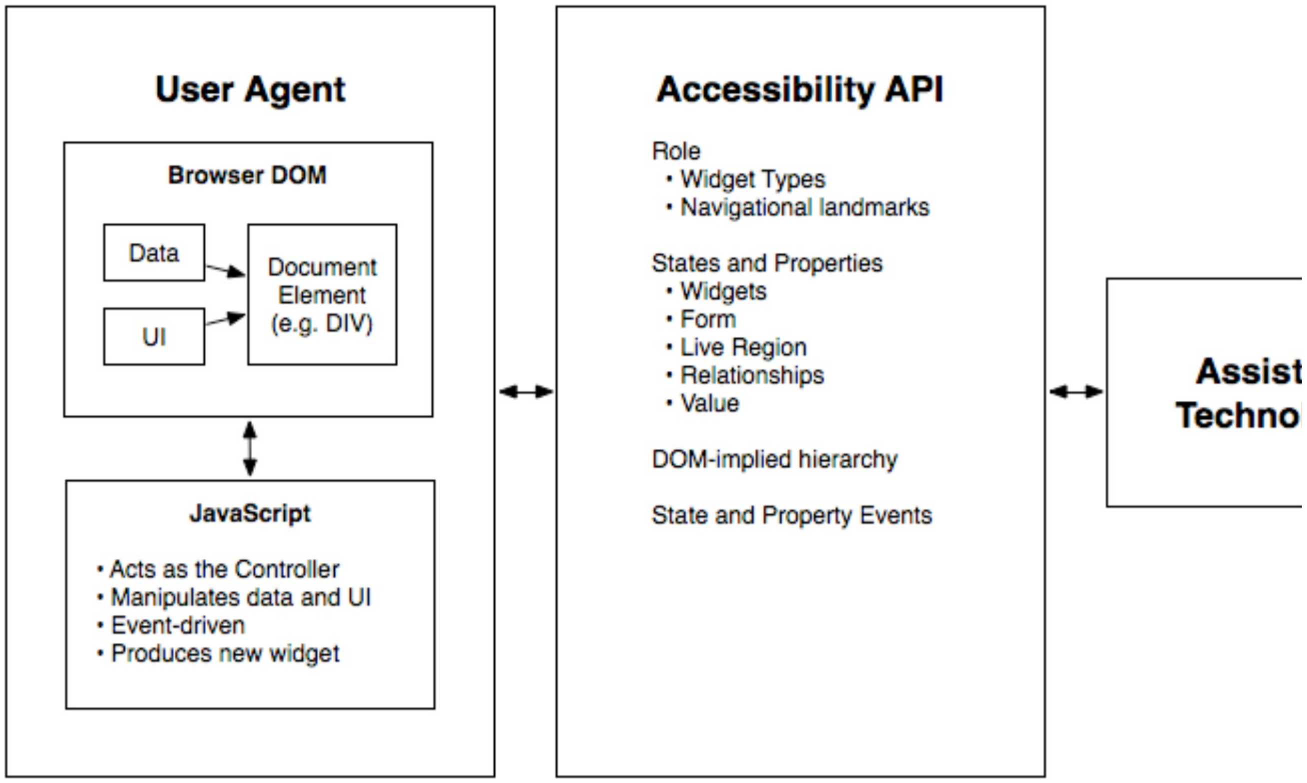


Figure 1: The contract model with accessibility APIs

For more information see the *WAI-ARIA Primer* [*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*], which is an implementation of *Resource Description Framework (RDF)* [*RDF*]. Tools can use these to validate the implementation of roles in a given content document.

> 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 ARIA is expected to define how to extend roles.

1.1. Scope

The design aims of creating this specification include:

- Define what state information may be controlled by the author.
- Support device independence - new devices coming on line, such as telephones, PDA's, tablets, televisions, etc. mean that it is imperative to have a design that allows you to author once and render in different ways on different devices, rather than authoring new versions of the document for each type of device.
- Improve the accessibility of dynamic content such as that generated by script.
- Provide for interoperability with Assistive Technologies.

This draft currently handles two aspects of roles: GUI functionality and structural relationships of the element. For more information see the *WAI-ARIA Primer* [*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 *XML Schema* [*XSD*]. However, being able to define similarities between roles, such as baseConcepts and more descriptive definitions, would not be available in XSD. While this extensibility is possible, this version of the specification does not define how this extension is to be achieved.

WAI-ARIA is supported by a set of informative resources. In addition to the WAI-ARIA Roadmap, the *WAI-ARIA Primer* [*ARIA-PRIMER*] provides a basic introduction to the concepts behind and reason for ARIA, and the *WAI-ARIA Best Practices* [*ARIA-PRACTICES*] describe recommended usage patterns for Web content developers. These documents are intended to help authors learn the practice of putting WAI-ARIA to use.

1.2. Use Cases

Keyboard accessible content helps users of alternate input devices. The new semantics when combined by our style guide work will allow alternate input solutions to facilitate command and control via an alternate input solution.

Low vision solutions benefit from ARIA markup in that the improved keyboard navigation helps people with extremely low vision. Low vision solutions offer a degree of screen reading functionality (like AI Squared's Zoom text). Furthermore, ARIA introduces navigation landmarks both through our taxonomy and the XHTML role landmarks which dramatically improves keyboard navigation productivity. This is a huge benefit for alternate input solutions as well.

ARIA will also be used to assist people with cognitive impairments. The additional semantics will allow authors to restructure and substitute alternative content in adaptive Web 2.0 solutions.

Assistive technology needs the ability to support alternative input forms by getting and setting the current value of widgets. AT also needs to determine what objects are selected, and manage widgets that allow multiple selections.

ARIA is used as a supplement for native language semantics, not a replacement. When the host language provides a feature that is equivalent to the ARIA feature, use the host language feature. ARIA should only be used in cases where the host language lacks the needed role, state, or property indicator. First use a host language feature that is as similar as possible to the ARIA feature, then refine the meaning by adding ARIA. For instance, a multiselectable grid could be implemented as a table, and then ARIA used to clarify that it is a grid, not just a table. This allows for the best possible fallback for user agents that do not support ARIA and preserves the integrity of the host language semantics.

2. Using WAI-ARIA

This section is *informative*.

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 *Roadmap for Accessible Rich Internet Applications* [*ARIA-ROADMAP*]). ARIA divides the semantics into roles (the type defining a user interface element) and states and properties supported by the roles.

Authors must associate elements in the document to an ARIA role and the appropriate attributes (states and properties) during its lifecycle.

2.1. WAI-ARIA Roles

An ARIA role is set on an element using the a role attribute similar to the one defined in the *XHTML Role Attribute Module* [*XHTML-ROLES*].

```
<div role = "checkbox">
```

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

The Roles in this taxonomy were modeled using *RDF/OWL* [*OWL*] to build rich descriptions of the expected behaviors of each role. Features of the role taxononmy provide the following information for each role:

- describes the context of a role, for example a `listitem` should be inside a `list`;
- defines a hierarchy of roles with related properties, for example a `directory` is a type of `list`;
- makes use of OWL to provide a type hierarchy allowing for semantic inheritance similar to a class hierarchy;
- makes references to related concepts in other specifications;
- provides descriptions for each role;
- describes what states are supported for each role, for example, a `checkbox` supports being `checked`.

Attaching a role from the role taxonomy to an element in the document gives assistive technology the information it needs to handle an element correctly.

2.2. WAI-ARIA States and Properties

ARIA provides a collection of accessibility states and properties which are used to support platform accessibility APIs on the various operating system platforms. Assistive technology 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 technology with information to render the information to the user at any instance in time. Changes in states or properties will result in a notification to an assistive technology to let them know that a change in behavior has occurred.

In the following example, a `span` has been used to create a checkbox which in this case has three possible states. A role is used to make the behavior of this simple widget known to the user agent. Properties that may change with user actions (such as `checked`) are defined in the States and Properties.

```
<span role="checkbox" aria-checked="mixed"
    onkeydown="return checkBoxEvent(event);"
    onclick="return checkBoxEvent(event);" >
  A checkbox label
</span>
```

It should be noted that some accessibility state information is managed and controlled by the user agent. These are called managed states. Often these states have corresponding CSS pseudo-classes to reflect necessary style changes. An example of a managed state is `focus`. In contrast, the states in this specification are typically controlled by the author and are called unmanaged states. Some states are managed by the UA but the author can override them, such as `posinset` and `setsize`. The author should override those only if the DOM is not complete and thus the UA calculation would be incorrect. Both managed and unmanaged states are mapped to the platform accessibility APIs by the user agent.

Finally, an added value for ARIA is that user agents that support *CSS attribute selectors* ([*CSS*], Section 5.8) can allow the author to create UI changes based on the ARIA state information, dramatically reducing the amount of script used to create the equivalent functionality. In the following example, CSS selectors based on the value of the ARIA `checked` state are used to determine whether an image of a checked or an unchecked box is shown:

```
*[aria-checked=true]:before  {content:  url('checked.gif')}

*[aria-checked=false]:before {content:  url('unchecked.gif')}
```

2.3. Managing Focus

In any `application` there must always be an element with focus, as applications require users to have a place to provide user input. The element with focus must never be destroyed, hidden or scrolled off-screen. All interactive elements should be focusable. There should be an obvious, discoverable way, either through tabbing or other standard navigation techniques, for keyboard users to move the focus to any interactive element they wish to interact with. See User Agent Accessibility Guidelines, Guideline 9 ([*UAAG*], Guideline 9).

When using standard (X)HTML and basic WAI-ARIA widgets, application developers can simply manipulate the tab order or use script to create keyboard shortcuts to elements in the document. Use of more complex widgets requires the author to manage focus within them.

WAI-ARIA includes a number of "managing container" widgets, also often referred to as "composite" widgets. Typically, the container is responsible for tracking the last descendant which was active (the default is usually the first item in the container). When the container is navigated to with the Tab key, focus goes directly to the last active descendant. The user may also activate the container by clicking on one of the descendants within it.

When something in the container has focus, the user may navigate through the container by pressing additional keys such as the arrow keys to move relative to the current item. 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`s, all of which may not be present in the document at one time. This requires their focus to be managed by the container using the `activedescendant` property, 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 Best Practices* ([*ARIA-PRACTICES*], section 3.2).

Containers that manage focus in this way are:

- `combobox`
- `grid`
- `listbox`
- `menu`
- `menubar`
- `tree`
- `treegrid`
- `tablist`
- `toolbar`

2.4. Features Managed by the User Agent

Editorial Note: Content is expected to come from the ARIA User Agent Implementors Guide. There are things like focused, focusable, selected, selectable, checkable, etc. Certain roles also have to be treated specially.

2.5. Building Accessible Applications with WAI-ARIA

This section provides a brief introduction to the process of making applications accessible using ARIA. The choice and usage of roles can be complex and context dependent. It is beyond the scope of this document to explain implementations for all the possible ARIA use cases. *ARIA Best Practices* [*ARIA-PRACTICES*] provides detailed guidance on ARIA implementation methodology as well as references to sample code.

First steps to making an application accessible:

1. Each element or widget has full and correct semantics that fully describes its behavior (using element names or roles);
2. The relationships between elements and groups are known;
3. States, properties, and container relationships are valid for each element's behavior and are accessible via the *Document Object Model* [*DOM*] and the platform accessibility API;
4. There is an element with input focus.

ARIA provides authors with the means to make the different elements in a Web application semantically rich. User agents use the role semantics to understand how to handle each element. Roles convey missing information that the assistive technology needs to anticipate the behavior of the elements inside the application such as how to present the corresponding ARIA states and properties to the user. The user agent will use the accessibility semantics from the host language and ARIA accessibility semantics which may override those of the host language and present them to an assistive technology through the Document Object Model or the platform accessibility API. When supporting the platform accessibility API the user agent will create accessible objects containing the accessibility semantics for each visual element on your page. It will use the chosen API to notify the assistive technology of changes to the semantics as well.

The following steps are recommended as ARIA is applied to content:

1. **Use native markup when possible**

   Use the semantic elements that are defined in the host markup language. For example, with XHTML it is better to use the native checkbox than to use a div element with role `checkbox` as these should already be accessible through your browser. There may also be cases where ARIA can augment an existing element in the host language. For example, a `grid` and `gridcell`s can reuse the functionality of a table when overlaying it. ARIA roles, states, and properties are best used when the markup language does not support all the semantics required. When a role attribute is added to an element, the semantics and behavior of the element are overridden by the role behavior.

2. **Apply the appropriate roles from ARIA**

   Set roles to make sure elements behave predictably and correctly describe the behavior of each element within the application (unless elements behaviors are fully described by the native markup language). Roles for interactive elements should support all the states that the element could use. Once a role attribute is set it should not be changed as this could confuse an assistive technology. This does not preclude an element being removed which has the role attribute set. Only states and properties may be changed for a given element.

3. **Preserve semantic structure**

   Structural information is critical to providing context to people with disabilities. This is achieved through preserving DOM hierarchy within structural elements and widgets; forming logical groups within within user interface widgets such as `treeitem`s in a `group`. Look for groups within a page, and mark them using the most appropriate role that best describes their usage. For example: a region of the page that contains a group of elements that are likely to change through an Ajax application could be tagged as a `region`. The preservation of semantic web structure involves your entire web page such as through the use of document landmarks to facilitate keyboard navigation for screen reader and mobility impaired users as well as page restructuring and simplification for users with cognitive and learning impairments.

4. **Build relationships**

   Look for relationships between elements, and mark them using the most appropriate property or attribute. For example: If container A contains search results, and container B contains the search widgets, then mark each container as a `region` and set the `controls` property in region B to reference region A. See relationships in WAI-ARIA.

Some relationships are determined automatically from the host language, such as by using the `label` tag in HTML.

5. **Set and properties in response to events**

   Once the role for an element has been set, select the appropriate states and properties for that role during the element's life cycle. This is often done in response to user input events. States are special properties for widgets that may change frequently during a widget's life cycle due to user interaction, while properties are more stable attributes of objects. User agents should notify assistive technology of state changes. Conversely, assistive technology notification of property changes depends on the method by which an assistive technology communicates with the user agent. For example, the `multiline` property is not something that changes frequently, whereas the `checked` state changes frequently in response to user input.

   When setting states and properties, set them until the behavior of the element is fully defined. Only use those supported for the chosen role or element as defined in this specification.

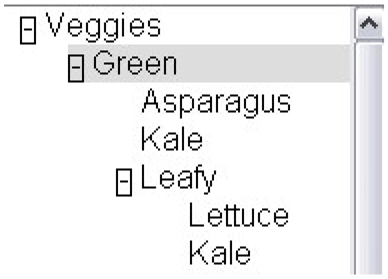
6. **Support full, usable keyboard navigation**

   Usable keyboard navigation in a rich internet application is different from the tabbing paradigm in a static document. Rich internet applications behave more like desktop applications where the user tabs to significant widgets and uses the arrow keys to navigate within the widget, such as a spreadsheet or menu. The changes that ARIA introduces in keyboard navigation makes this enhanced accessibility possible. For a more in-depth understanding of keyboard navigation in ARIA, see the *ARIA Best Practices* [*ARIA-PRACTICES*]

7. **Synchronize the visual UI with accessibility states properties for supporting user agents**

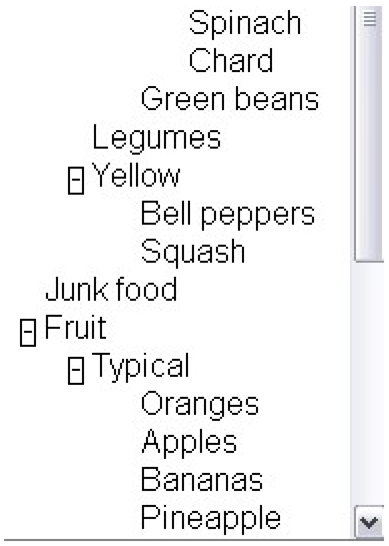
   This will allow the state of your UI to be perceivable to the user as well as the assistive technology. There are many ways to do this using script or by using CSS attribute selectors (in conforming user agents) For example, the author should have an associated selector that responds to a form element being `required` or a gridcell being `selected`. Refer to the *ARIA Best Practices* [*ARIA-PRACTICES*] for techniques for proper UI synchronization with the accessible state of the document.

2.6. Example: building a tree widget



A basic tree view allows the user to select different list items and expand and collapse embedded lists. Arrow keys are used to navigate through a tree, including left/right to collapse/expand sub trees. Double clicking with the mouse also toggles expansion.

Building this user interface element with script could leave



assistive technology guessing about the role of each element. To make this feature accessible we need to:

- Inform assistive technology about the role of each element, so they know how to handle it;
- Inform assistive technology about the relationships between tree items;
- Give a clear keyboard focus that will not confuse users with disabilities;
- Expose the changing states (e.g., open and closed) of the the tree items.

We can do that by following the steps below:

1. **Look at the native mark up language**

   There is no tree element in HTML that supports our behavior including expansion. If such an element existed, we should use that to take advantage of existing support. Since it does not, we will need to use roles.

2. **Finding the right roles**

   As we did not find a tree element in the native mark up we need to add roles that do this by referencing roles from this taxonomy that support states that we need.

   Our tree will need roles that support embedded list behavior and expandable/collapsible embedded lists. The roles that support tree behavior for a tree are:

   - `tree`: A tree is the main container element for our tree. It is a form of a `select` where subtrees can be collapsed and expanded.
   - `treeitem`: A treeitem is an option item of a tree. This is an element within a tree which may be expanded or collapsed.

3. **Look for groups and build relationships**

   Tree relationships can be made simply via the DOM and logical structure of the page. A tree element will be the main container encompassing all other elements in the tree. Each selectable item in the tree will be a treeitem.

   When a treeitem contains an embedded list of treeitems they will be all embedded in a group. A group should be contained inside the tree item that is the parent item.

   Tree relationships are like list relationships in HTML. Group and tree elements act like list containers (`ol` and `ul`). A tree item acts like a list item (`li`) in HTML.

   Because treeitems and groups commonly both use `div` elements it is recommended to add a comment next to closing treeitems that contain embedded tree groups.

```
<div role="tree">
 <div role="treeitem">Veggies</div>
 <div role="group">   <!-- veggies children ->
   <div role="treeitem">Green</div>
   <div role="group">  <!-- green children -->
     <div role="treeitem">Asparagus</div>
     <div role="treeitem">Kale</div>
     <div role="treeitem">Leafy</div>
     <div role="group">   <!-- leafy children -->
       <div role="treeitem">Lettuce</div>
       <div role="treeitem">Kale</div>
       <div role="treeitem">Spinach</div>
       <div role="treeitem">Chard</div>
     </div>   <!-- close leafy -->
     <div role="treeitem">Green beans</div>
    </div>  <!-- close green -->
   <div role="treeitem">Legumes</div>
   <div role="treeitem">Yellow</div>
   <div role="group">   <!-- yellow children -->
     <div role="treeitem">Bell peppers</div>
     <div role="treeitem">Squash</div>
   </div>  <!-- close yellow -->
  </div>  <!-- close veggies -->
</div>  <-- close tree -->
```

Sometimes a tree structure is not explicit via the DOM and logical structure of a page. In such cases the relationships must still be made explicit using the states and properties. In the following example, the `owns` property indicates that the item with id "yellowtreegroup" should be considered a child of the `div` element with the property, even though it is not a child in the DOM.

```
<div role="treeitem" aria-owns="yellowtreegroup">Yellow<div>
…
<div id="yellowtreegroup" role="group">
<div role="treeitem">Bell peppers</div>
<div role="treeitem">Squash</div>
…
</div>
```

If the tree is not completely represented in the DOM at all times, don't use either the structured or `owns` methods. Instead use `level`, `posinset` and `setsize`.

4. **Use States, Properties in response to events**

   Control the behavior of the element in response to user input events such as from the keyboard and the mouse as shown here:

```
<div tabindex="-1" role="treeitem" aria-expanded="true"
    onclick="return toggleExpansion(event)"
    onkeydown="return processArrowKeystoToggleExpansion(event);"
>Yellow</div>
```

   Use device independent events with supporting JavaScript to handle user interaction:

```
<div role="tree" tabindex="-1"
    onfocus="return treeItemFocus(event);"
    onclick="return treeItemEvent(event);"
    ondblclick="return treeItemEvent(event);"
    onkeydown="return treeItemEvent(event);">
```

Create JavaScript support to control the event driven behavior of the application.

3. Normative Requirements

This section is *normative.*

This specification indicates whether a section is *normative* or *informative*.

Normative sections provide requirements that must be followed for an implementation to conform to this specification. The keywords **MUST**, **MUST NOT**, **REQUIRED**, **SHALL**, **SHALL NOT**, **SHOULD**, **RECOMMENDED**, **MAY**, and **OPTIONAL** in this document are to be interpreted as described in *Key words for use in RFCs to indicate requirement levels* [*RFC2119*].

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.

4. The Roles Model

This section is *normative.*

This section defines the WAI-ARIA role taxonomy and describes the characteristics and properties of all roles. A formal RDF representation of all the information presented here is available in Appendix 8.1: Implementation.

4.1. Relationships between concepts

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

4.1.1. Parent Roles

RDF Property
rdfs:subClassOf

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

For example: `checkbox` is a subclass or type of a `option`. If we change the properties and expected behavior of an `option` then the properties and behavior of `checkbox` will also change.

Inheritance is expressed in RDF using the RDF Schema *subClassOf* ([*RDFS*], section 3.4) property.

4.1.2. Child Roles

RDF Property
<none>

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

4.1.3. Related Concepts

RDF Property
role:relatedConcept

Informative information 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 a type changes, the properties, behavior and definition of a related concept is not affected.

For example: A grid is like a table. Therefore, a grid has a `rdfs:seeAlso` of table. However if the definition of `table` is modified, the ARIA definition of a grid will not be affected.

4.1.4. Base Concept

RDF Property
role:baseConcept

Informative information 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 relatedConcept except that base concepts are almost identical to each other.

For example: the `checkbox` defined in this document has the same functionality and anticipated behavior as a checkbox defined in HTML.

Therefore, a `checkbox` has an HTML `checkbox` as a `baseConcept`. However, if the HTML checkbox is modified, the definition of a `checkbox` in this document will not be not affected, because there is no actual inheritance of type.

4.2. 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, what instances of 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.

4.2.1. Is Abstract

RDF Property
    N/A
Values
    Boolean

Abstract roles are the foundation upon which all other ARIA roles are built. They **MUST NOT** be used by authors because they are not implemented in the API binding. 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 appropriate features.

4.2.2. Required States and Properties

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

States and properties required for the role. Content authors **MUST** provide values 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.

4.2.3. Supported States and Properties

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

States and properties applicable to the role. User agents **MUST** support all supported states and properties for the role. Content authors **MAY** provide values for supported states and properties, but may not in all cases because default values are sufficient.

4.2.4. Inherited States and Properties

Informative list of properties that are inherited onto a role from ancestor roles. States and properties are inherited from ancestor 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. Inherited states and properties that are required are indicated as such in this field as well. 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.

4.2.5. Required Child Elements

RDF Property
role:mustContain

Values
Any valid RDF object reference, such as a URI or RDF ID reference.

A child element that must be contained in the DOM by this role. A child element is any descendent element specified with the required child role. For example, an element with role `list` must contain an element with role `listitem`.

When multiple required children are indicated, either of them are permitted.

4.2.6. Parent Element

RDF Property
role:scope

Values
Any valid RDF object reference, such as a URI or RDF ID reference.

Context where this role is allowed, in other words, roles for elements in which the present role **MUST** appear.

For example an element with role `listitem` **MUST** be contained inside an element with role `list`.

4.2.7. Name From

RDF Property
role:nameFrom

Values
One of the following values:

- "author": name comes from values provided by the author in explicit markup features such as the `labelledby` property or the HTML "title" attribute.
- "subtree": name comes from the text value of the element node. Although this may be allowed in addition to "author" in some roles, this is used in content only if "author" features are not provided.

Computational mechanism to determine the accessible name of the object to which the role applies. This may be computed from the descendants of the object or conditional text (e.g., the `title` attribute in HTML).

User agents **MUST** use the following approach to compute the accessible name:

Collect the name from the content subtrees pointed to by `labelledby` which contains the IDs for the label content. Use the IDs in the order they appear. For each ID, use a depth-first computation of the name, appending to the currently computed name.

If `labelledby` is unspecified:

1. For container elements which allow name from descendants, use a depth-first computation of the name.
    1. If a text node, append the text contents;
    2. If a widget, append the current value for the widget;

    3. If an image or object with a text equivalent, append the text equivalent;
    4. If there is a forced line break (e.g. if the current object is styled with display:block), append a space character.
2. If not such a container, or the descendant computation generated an empty name, use the contents of conditional text, if specified.

4.2.8. Children Presentational

RDF Property
: role:childrenArePresentational

Values
: Boolean (true | false)

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

4.3. Global States and Properties

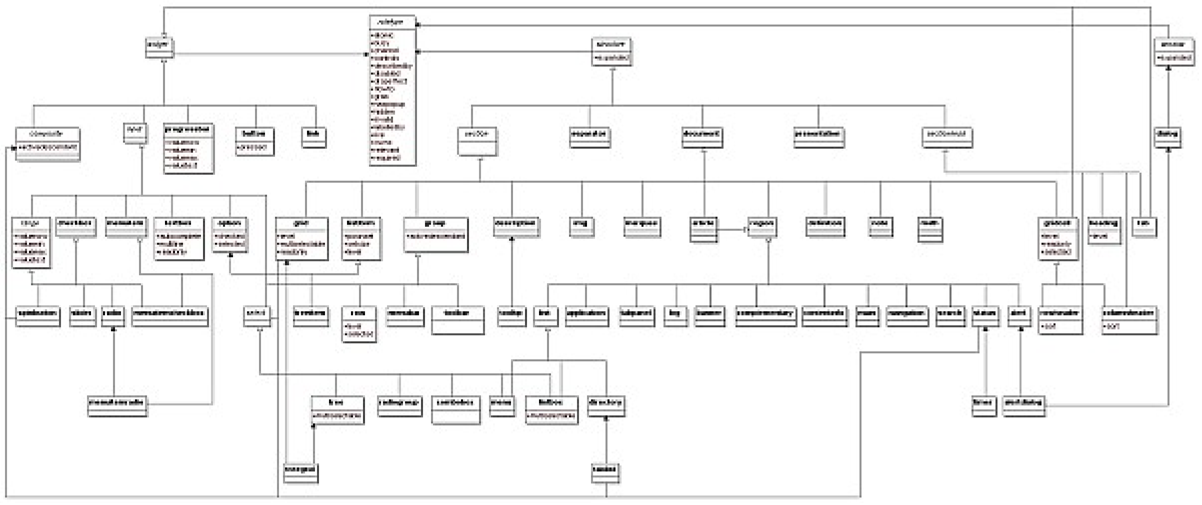
Some states and properties are applicable to all roles, and most are applicable to all elements regardless of role. In addition to explicitly expressed supported states and properties, the following global states and properties are supported by all roles as well. These include:

- `atomic`
- `busy`
- `channel`
- `controls`
- `describedby`
- `disabled`
- `dropeffect`
- `flowto`
- `grab`
- `haspopup`
- `hidden`
- `invalid`
- `labelledby`
- `live`
- `owns`
- `relevant`
- `required`

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.

4.4. Roles

To support the current user scenario, this specification defines roles that A, help define Widgets (For example, a tristate Checkbox) and B, help define page structure (for example, a section header).



Class diagram of the relationships described in the role data model

View larger version of class diagram

Roles are categorized as follows:

1. Taxonomy Roles
2. User Input Controls
3. User Interface Elements
4. Document Structure
5. Specialized Regions

Below is an alphabetical list of ARIA roles to be used by rich internet application authors (excluding abstract roles which are not used directly). A detailed definition of the taxonomy supporting these ARIA roles follows.

`alert`
: A message with important information.

`alertdialog`
: A separate window (may be simulated) with an alert, where initial focus goes to the window or a widget within it.

`application`
: A software unit executing a set of tasks for its users.

`article`
: Represents a section of a page consisting of a composition forming an independent part of a document, page, or site.

`banner`
: A region that contains the prime heading or internal title of a page.

`button`
: Allows for user-triggered actions.

`checkbox`
: A widget that has three possible values: "true", "false", or "mixed".

`columnheader`
: A table cell containing header information for a column.

`combobox`	Combobox is a presentation of a select, where users can type to select an item.
`complementary`	Any section of the document that supports but is separable from the main content, but is meaningful on its own even when separated from it.
`contentinfo`	Meta information which applies to the first immediate ancestor whose role is not presentation.
`definition`	A definition of a term or concept.
`description`	Descriptive content for a page element.
`dialog`	A dialog is a small application window that sits above the application and is designed to interrupt the current processing of an application in order to prompt the user to enter information or require a response.
`directory`	A list of references to members of a single group.
`document`	Content that contains related information.
`grid`	A grid contains cells of tabular data arranged in rows and columns (e.g., a table).
`gridcell`	A cell in a grid.
`group`	A section of user interface objects which would not be included in a page summary or table of contents by an assistive technology.
`heading`	A heading for a section of the page.
`img`	A container for a collection of elements that form an image.
`link`	Interactive reference to a resource.
`list`	Group of non-interactive list items.
`listbox`	A widget that allows the user to select one or more items from a list of choices.
`listitem`	A single item in a list.
`log`	A region where new information is added and old information may disappear.
`main`	Main content in a document.
`marquee`	A marquee is used to scroll text across the page.
`math`	Content that represents a mathematical expression.
`menu`	Offers a list of choices to the user.
`menubar`	A container of menu items (items with role menuitem).

`menuitem`	An option in a group of choices contained in a menu.
`menuitemcheckbox`	Defines a menuitem which is checkable.
`menuitemradio`	Indicates a menuitem which is part of a group of menuitemradio roles, only one of which can be checked at a time.
`navigation`	A collection of links suitable for use when navigating the document or related documents.
`note`	The content is parenthetic or ancillary to the main content of the resource.
`option`	A selectable item in a list represented by a select.
`presentation`	An element whose role is decorative, not meaningful, and does not need to be mapped to the accessibility API.
`progressbar`	Displays the execution status for tasks that take a long time to execute.
`radio`	An option in single-select list.
`radiogroup`	A group of radio buttons.
`region`	Region is a large perceivable section on the web page.
`row`	A row of grid cells.
`rowheader`	A table cell containing header information for a row.
`search`	The search tool of a web document.
`separator`	A line or bar that separates and distinguishes sections of content.
`slider`	A user input where the user selects a value from within a given range.
`spinbutton`	A form of range that expects a user to select from amongst discrete choices.
`status`	Container for processing advisory information to give feedback to the user.
`tab`	A header for a tabpanel.
`tablist`	A list of tabs, which are references to tabpanels.
`tabpanel`	A container for the resources associated with a tab.
`textbox`	Inputs that allow free-form text as their value.
`timer`	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 functions represented in compact visual form.

tooltip
: A popup that displays a description for an element when a user passes over or rests on that element. Supplement to the normal tooltip processing of the user agent.

tree
: A form of a list having groups inside groups, where sub trees 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.

4.4.1. Base Types

The following roles are used as base types for applied roles. Base classes are used to build a picture of the role taxonomy class hierarchy within the taxonomy. Note that while all roles in this section are abstract, not all abstract roles are in this section. This section includes only the most high-level abstractions in the taxonomy.

Roles in this section include:

- roletype
- widget
- structure
- composite
- window

Role: `roletype`

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: This is used for the ontology and authors **MUST NOT** use this role in content.

Characteristics:

Characteristic	**Value**
Is Abstract:	True
Child Roles:	widget structure window

Characteristic	Value
Related Concepts:	XHTML role HTML link (rel & rev) Dublin Core type

Supported States and Properties:	`atomic`
	`busy`
	`channel`
	`controls`
	`describedby`
	`disabled`
	`dropeffect`
	`flowto`
	`grab`
	`haspopup`
	`hidden`
	`invalid`
	`labelledby`
	`live`
	`owns`
	`relevant`
	`required`

Role: `widget`

A component of a Graphical User Interface (GUI).

Widgets are discrete user interface elements with which the user can interact. Widget roles all map to standard features in accessibility APIs.

> Note: This is used for the ontology and authors **MUST NOT** use this role in content.

Characteristics:

Characteristic	Value
Is Abstract:	True
Parent Roles:	`roletype`

Child Roles:	`composite`
	`input`
	`button`
	`link`
	`tab`
	`gridcell`
	`progressbar`

Inherited States and Properties:	Global States

Role: `structure`

A document structural element.

Roles for document structure support the accessibility of dynamic Web content by helping assistive technology to determine active content vs. 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.

The taxonomy of structural roles is likely to evolve as new use cases are added to the scope of this specification.

> Note: This is used for the ontology and authors **MUST NOT** use this role in content.

Characteristics:

Characteristic	Value
Is Abstract:	True
Parent Roles:	`roletype`
Child Roles:	`section` `sectionhead` `document` `presentation` `separator`
Supported States and Properties:	`expanded`
Inherited States and Properties:	Global States

Role: `composite`

A widget that may contain navigable descendants or owned children.

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

Descendants of this role **MUST NOT** have the "nameFrom" value of "subtree" set. This role and its descendants **MUST NOT** have a childrenArePresentational value of "true".

> Note: This is used for the ontology and authors **MUST NOT** use this role in content.

Characteristics:

Characteristic	Value
Is Abstract:	True
Parent Roles:	`widget`

Characteristic	Value
Child Roles:	select spinbutton tablist grid status
Supported States and Properties:	activedescendant
Inherited States and Properties:	Global States
Name From:	author
Children Presentational:	False

Role: `window`

Browser or application window.

Elements with this role have a window-like behavior in a GUI context, regardless of whether they are implemented as a window in the OS. This is helpful when there is the visual appearance of a window that is merely a styled section of the document.

Note: This is used for the ontology and authors **MUST NOT** use this role in content.

Characteristics:

Characteristic	Value
Is Abstract:	True
Parent Roles:	roletype
Child Roles:	dialog
Supported States and Properties:	expanded
Inherited States and Properties:	Global States
Name From:	author

4.4.2. User Input Widgets

These roles are common widgets used to collect and maintain user input.

Roles in this section include:

- `input`
- `select`
- `listbox`
- `combobox`
- `option`
- `checkbox`
- `radiogroup`
- `radio`

- `textbox`
- `range`
- `slider`
- `spinbutton`

Role: `input`

Generic type for widgets for which users can input a value.

Characteristics:

Characteristic	Value
Is Abstract:	True
Parent Roles:	`widget`
Child Roles:	`select` `option` `checkbox` `textbox` `range` `menuitem`
Related Concepts:	XForms input
Inherited States and Properties:	Global States
Name From:	author

Role: `select`

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

An element with role `select` **MUST** contain elements with role `option`.

> Note: This is used for the ontology and authors **MUST NOT** use this role in content.

Characteristics:

Characteristic	Value
Is Abstract:	True
Parent Roles:	`composite` `group` `input`
Child Roles:	`listbox` `combobox` `radiogroup` `menu` `tree`
Inherited States and Properties:	`activedescendant` `expanded`

Characteristic	Value
	Global States
Name From:	author

Role: `listbox`

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

Items within the list are static and may contain images. List boxes contain children whose role is `option`.

Instances of this role **MUST** manage focus of descendants, as described in Managing Focus.

Although `listbox` inherits the `expanded` state, if there is a use case to use that state, authors should consider using a `combobox` instead.

Characteristics:

Characteristic	Value
Parent Roles:	`list` `select`
Related Concepts:	HTML select XForms select
Required Child Elements:	`option`
Supported States and Properties:	`multiselectable`

Inherited States and Properties:	`activedescendant`
	`expanded`
	Global States

Name From:	author
Accessible Name Required:	True

Role: `combobox`

Combobox is a presentation of a `select`, where users can type to select an item.

Combobox is the combined presentation of a single line textbox with a drop down select widget. The combobox may be editable. Typically editable combo boxes are used for autocomplete behavior, and the autocomplete property would be used on the child textbox.

> NOTE: In *XForms* [*XFORMS*] the same `select` can have one of 3 appearances: combo-box, drop-down box, or group of radio-buttons. Many browsers (if not all of them) allow users to type in a drop-down select as well. This specification does not constrain the presentation of the

combobox.

Instances of this role **MUST** manage focus of descendants, as described in Managing Focus.

Characteristics:

Characteristic	**Value**
Parent Roles:	`select`
Related Concepts:	HTML select XForms select
Required Child Elements:	`listbox`
Required States and Properties:	`expanded`
Inherited States and Properties:	`activedescendant` Global States
Name From:	author
Accessible Name Required:	True

Role: `option`

A selectable item in a list represented by a `select`.

An option **MUST** appear inside an element with `select` role.

Characteristics:

Characteristic	**Value**
Parent Roles:	`input`
Child Roles:	`treeitem`
Base Concept:	HTML option
Related Concepts:	`listitem` XForms item
Parent Element:	`select`
Supported States and Properties:	`checked` `selected`
Inherited States and Properties:	Global States
Name From:	subtree author
Accessible Name Required:	True

Role: `checkbox`

A widget that has three possible values: "true", "false", or "mixed".

Many checkboxes do not use the "mixed" value, and thus are effectively boolean checkboxes. However, the `checked` state supports the "mixed" value to support cases such as installers where an option has been partially installed.

Characteristics:

Characteristic	**Value**
Parent Roles:	`input`
Child Roles:	`radio` `menuitemcheckbox`
Related Concepts:	HTML input (type : checkbox) `option`
Required States and Properties:	`checked`
Inherited States and Properties:	Global States
Name From:	subtree author
Accessible Name Required:	True

Role: `radiogroup`

A group of `radio` buttons.

A `radiogroup` is a type of `select` list that can only have single entries checked, not mutliple. User agents **MUST** enforce that only one radio button in a radiogroup can be checked at the same time. When another button is checked, previously checked buttons become unchecked (their `checked` state becomes "false").

Characteristics:

Characteristic	**Value**
Parent Roles:	`select`
Related Concepts:	`list`
Required Child Elements:	`radio`

Inherited States and Properties:	`activedescendant`
	`expanded`
	Global States

Name From:	author
Accessible Name Required:	True

Role: `radio`

An option in single-select list.

Elements with role `radio` **MUST** be be explicitly grouped in order to indicate which ones affect the same value. This should be done by enclosing them in an element with role `radiogroup`. If it is not possible to make the radio buttons DOM children of the `radiogroup`, use the `owns` property on the `radiogroup` element to indicate the relationship.

Characteristics:

Characteristic	Value
Parent Roles:	`checkbox`
Child Roles:	`menuitemradio`
Related Concepts:	HTML input (type : radio)
Inherited States and Properties:	`checked` **(required)** Global States
Name From:	subtree author
Accessible Name Required:	True

Role: `textbox`

Inputs that allow free-form text as their value.

If the `multiline` property is true, the widget accepts line breaks within the input, as in a HTML textarea. Otherwise this is a simple text box.

Intended use is in languages that do not have a text input object (such as SVG), or cases in which an element with different semantics is repurposed as an input box. Another use is for a rich text edit textbox, or one that validates input as users type (perhaps marking specific subregions with errors with the `invalid` state). Content authors **MAY** also use this role when additional states or properties are applied to a standard text input widget. This is to indicate to the user agent that it must process additional states and properties such as `invalid` and `required`.

Characteristics:

Characteristic	Value
Parent Roles:	`input`
Related Concepts:	XForms input HTML textarea

Supported States and Properties:	`autocomplete`
	`multiline`
	`readonly`

Inherited States and Properties:	Global States
Name From:	author
Accessible Name Required:	True

Role: `range`

Represents a range of values that can be set by the user.

> Note: This is used for the ontology and authors **MUST NOT** use this role in content.

Characteristics:

Characteristic	**Value**
Is Abstract:	True
Parent Roles:	`input`
Child Roles:	`slider` `spinbutton`
Supported States and Properties:	`valuemax` `valuemin` `valuenow` `valuetext`
Inherited States and Properties:	Global States
Name From:	author

Role: `slider`

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.

Characteristics:

Characteristic	**Value**
Parent Roles:	`range`
Required States and Properties:	`valuemax` `valuemin` `valuenow`
Inherited States and Properties:	`valuetext` Global States
Name From:	author
Accessible Name Required:	True
Children Presentational:	True

Role: `spinbutton`

A form of `range` that expects a user to select from amongst 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. This functionality **SHOULD** be accomplished programmatically through the use of up and down arrows on the keyboard.

Characteristics:

Characteristic	Value
Parent Roles:	`composite` `range`
Required States and Properties:	`valuemax` `valuemin` `valuenow`
Inherited States and Properties:	`activedescendant` `valuetext` Global States
Name From:	author
Accessible Name Required:	True

4.4.3. User Interface Elements

These roles encompass features that usually are used as part of the graphical user interface.

Roles in this section include:

- `button`
- `link`
- `menu`
- `menubar`
- `toolbar`
- `menuitem`
- `menuitemcheckbox`
- `menuitemradio`
- `tooltip`
- `tabpanel`
- `tablist`
- `tab`
- `tree`
- `treeitem`

Role: `button`

Allows for user-triggered actions.

Buttons are mostly used for discrete, atomic actions. Standardizing the appearance of buttons enhances recognition as buttons and arraying them compactly in toolbars, for example.

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

Characteristics:

Characteristic	Value
Parent Roles:	`widget`
Base Concept:	HTML button
Related Concepts:	`link` XForms trigger
Supported States and Properties:	`pressed`
Inherited States and Properties:	Global States
Name From:	subtree author
Accessible Name Required:	True
Children Presentational:	True

Role: `link`

Interactive reference to a resource.

Activating the link causes the user agent to navigate to that resource.

Characteristics:

Characteristic	Value
Parent Roles:	`widget`
Related Concepts:	HTML link
Inherited States and Properties:	Global States
Name From:	subtree author
Accessible Name Required:	True

Role: `menu`

Offers a list of choices to the user.

A menu is often a list of links to important sections of a document or a site. The `menu` role is appropriate when the list of links is presented in a manner similar to a menu on a desktop application.

Instances of this role **MUST** manage focus of descendants, as described in Managing Focus.

Characteristics:

Characteristic	Value
Parent Roles:	`list` `select`
Related Concepts:	DTB sidebar XForms select JAPI MENU

Required Child Elements:	`menuitem`
	`menuitemcheckbox`
	`menuitemradio`
	`separator`

Inherited States and Properties:	`activedescendant`
	`expanded`
	Global States

Name From:	author
Accessible Name Required:	True

Role: `menubar`

A container of menu items (items with role `menuitem`).

The `menubar` role is used to create a menubar similar to those found in Windows, the Mac, and Gnome desktops. A menubar is used to create a consistent climate of frequently used commands. Navigation behavior **SHOULD** be similar to the typical menu bar graphical user interface.

Instances of this role **MUST** manage focus of descendants, as described in Managing Focus.

Characteristics:

Characteristic	Value
Parent Roles:	`group`
Related Concepts:	`toolbar`

Inherited States and Properties:	`activedescendant`
	`expanded`
	Global States

Name From:	author

Role: `toolbar`

A collection of commonly used functions represented in compact visual form.

The toolbar is often a subset of functions found in a `menubar`, designed to reduced user effort in using these functions.

If this is not keyboard accessible the actions defined in the toolbar **MUST** be reproduced in an accessible, device independent fashion.

Instances of this role **MUST** manage focus of descendants, as described in Managing Focus.

Characteristics:

Characteristic	**Value**
Parent Roles:	`group`
Related Concepts:	`menubar`

Inherited States and Properties:	`activedescendant`
	`expanded`
	Global States

Name From:	author

Role: `menuitem`

An option in a group of choices contained in a `menu`.

A menuitem **MAY** be `disabled`. Each menu item **MAY** activate a new sub-menu, which is indicated with the `haspopup` property.

Characteristics:

Characteristic	**Value**
Parent Roles:	`input`
Child Roles:	`menuitemcheckbox` `menuitemradio`

Related Concepts:	JAPI MENU_ITEM
	`listitem`
	`option`

Parent Element:	`menu`
Inherited States and Properties:	Global States
Name From:	subtree author
Accessible Name Required:	True

Role: `menuitemcheckbox`

Defines a `menuitem` which is checkable.

The `checked` state indicates whether the menu item is checked, unchecked, or mixed.

Characteristics:

Characteristic	Value
Parent Roles:	`checkbox` `menuitem`
Related Concepts:	`menuitem`
Parent Element:	`menu`
Inherited States and Properties:	`checked` **(required)** Global States
Name From:	subtree author
Accessible Name Required:	True

Role: `menuitemradio`

Indicates a `menuitem` which is part of a group of `menuitemradio` roles, only one of which can be checked at a time.

User agents **MUST** enforce that only one `menuitemradio` in a group can be checked at the same time. When another widget is checked, previously checked widget become unchecked (their `checked` state becomes "false").

Menu items **SHOULD** be in an element with role `menu` in order to identify that they are related widgets, and **MAY** also be separated into a group by a `separator`, or an element with an equivalent role from the native markup language.

Characteristics:

Characteristic	Value
Parent Roles:	`menuitem` (see structure) `radio`
Related Concepts:	`menuitem`
Parent Element:	`menu`
Inherited States and Properties:	`checked` **(required)** Global States
Name From:	subtree author
Accessible Name Required:	True

Role: `tooltip`

A popup that displays a description for an element when a user passes over or rests on that element. Supplement to the normal tooltip processing of the user agent.

Objects with this role **SHOULD** be referenced through the use of `describedby`, at latest by the time the tooltip is displayed.

Characteristics:

Characteristic	**Value**
Parent Roles:	`description`
Inherited States and Properties:	`expanded` Global States
Accessible Name Required:	True

Role: `tabpanel`

A container for the resources associated with a `tab`.

> Note: There **MUST** be a means to associate a tabpanel element with its associated `tab` in a `tablist`. Using the `labelledby` property on the tabpanel to reference the tab is the recommended way to achieve this.

For detailed information about how to use tab panels, see the TabPanel Widget in WAI-ARIA Best Practices ([ARIA-PRACTICES], Section 9.2).

Characteristics:

Characteristic	**Value**
Parent Roles:	`region`
Inherited States and Properties:	`expanded` Global States
Name From:	author
Accessible Name Required:	True

Role: `tablist`

A list of `tab`s, which are references to `tabpanel`s.

Instances of this role **MUST** manage focus of descendants, as described in Managing Focus.

Characteristics:

Characteristic	**Value**
Parent Roles:	`composite` `directory`
Related Concepts:	DAISY Guide
Required Child Elements:	`tab`

Characteristic	Value
Inherited States and Properties:	`activedescendant` `expanded` Global States
Name From:	author

Role: `tab`

A header for a `tabpanel`.

`tab` is used as a grouping label, providing a link for selecting the tab content to be rendered to the user. If the `tab` or an object in the associated `tabpanel` has focus, the `tab` is the active one in the list.

One (and only one) of the `tab`s in the `tablist` **MUST** be the current tab. The `tabpanel` associated with the current tab **MUST** be rendered to the user. Other tabpanels **SHOULD** be hidden from the user until the user selects the tab associated with that tabpanel.

User agents manage the determination and indication of the current tab. There is no property in the taxonomy for this.

Characteristics:

Characteristic	Value
Parent Roles:	`sectionhead` `widget`
Parent Element:	`tablist`
Inherited States and Properties:	`expanded` Global States
Name From:	subtree author

Role: `tree`

A form of a `list` having groups inside groups, where sub trees can be collapsed and expanded.

Instances of this role **MUST** manage focus of descendants, as described in Managing Focus.

Characteristics:

Characteristic	Value
Parent Roles:	`select`
Child Roles:	`treegrid`

Characteristic	Value
Required Child Elements:	treeitem
Supported States and Properties:	multiselectable
Inherited States and Properties:	activedescendant expanded Global States
Name From:	author
Accessible Name Required:	True

Role: `treeitem`

An option item of a `tree`. This is an element within a tree that may be expanded or collapsed.

A collection of treeitems to be expanded and collapsed are enclosed in a `group`.

Characteristics:

Characteristic	Value
Parent Roles:	listitem option
Parent Element:	tree
Inherited States and Properties:	checked expanded level posinset selected setsize Global States
Name From:	subtree author
Accessible Name Required:	True

4.4.4. Document Structure

These roles describe the structures that organize content in a page. In contrast to widgets, structures are not usually interactive. However, they can be in certain circumstances.

Roles in this section include:

- `section`
- `sectionhead`
- `document`
- `region`
- `heading`

- `list`
- `listitem`
- `group`
- `grid`
- `row`
- `gridcell`
- `rowheader`
- `columnheader`
- `treegrid`
- `description`
- `directory`
- `img`
- `presentation`
- `separator`
- `math`

Role: `section`

A renderable structural containment unit in a document or application.

Note: This is used for the ontology and authors **MUST NOT** use this role in content.

Characteristics:

Characteristic	**Value**
Is Abstract:	True
Parent Roles:	`structure`

Child Roles:	`region`
	`listitem`
	`group`
	`grid`
	`gridcell`
	`description`
	`img`
	`math`
	`marquee`
	`definition`
	`note`

Related Concepts:	DTB frontmatter DTB level SMIL par
Inherited States and Properties:	`expanded` Global States
Name From:	subtree author

Role: `sectionhead`

Labels or summarizes the topic of its related section.

> Note: This is used for the ontology and authors **MUST NOT** use this role in content.

Characteristics:

Characteristic	**Value**
Is Abstract:	True
Parent Roles:	`structure`

Child Roles:	`tab`
	`heading`
	`rowheader`
	`columnheader`

Inherited States and Properties:	`expanded` Global States
Name From:	subtree author

Role: `document`

Content that contains related information.

The document role informs screen readers 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 withn a region with role="application", where all text should be semantically associated with focusable elements. An important trait of documents is that they have some text which is not associated with widgets or groups thereof.

To properly set the role of `document`, an author should set the `document` role on an element which encompasses the entirety of the region for which assistive technology browser navigation mode is applied.If it applies to the entire Web page, it should be set on the root note for content, e.g., `body` in HTML or `svg` in SVG.

Documents **MUST** have a document title or label. This should be suitable for use as a navigation preview or table-of-contents entry for the page section in question. The label **SHOULD** come from one of the following sources:

- The `title` element in HTML labels the entire document;
- Any `group` element in SVG may contain a non-empty `title` element;
- The element with the `document` role may have a `labelledby` property referencing one or more elements with non-empty text content.

Characteristics:

Characteristic	**Value**

Characteristic	Value
Parent Roles:	structure
Child Roles:	article
Related Concepts:	Device Independence Delivery Unit
Inherited States and Properties:	expanded Global States
Name From:	author
Accessible Name Required:	True

Role: `region`

Region is a large perceivable section on the web page.

This role defines a group of elements that together form a large perceivable section, that the author feels should be included in a summary of page features. A `region` **MUST** have a heading, provided via an instance of the `heading` role or using the `labelledby` property to reference an element. A region does not necessarily follow the logical structure of the content, but follows the perceivable structure of the page.

When defining regions of a web page, authors should consider using standard document landmark roles. If the definition of these regions are inadequate, authors should use the `region` role and provide the appropriate title text.

Characteristics:

Characteristic	Value
Parent Roles:	section
Child Roles:	tabpanel list application alert log status article banner complementary contentinfo main navigation search
Related Concepts:	HTML Frame Device Independence Glossary perceivable unit section

Characteristic	Value
Inherited States and Properties:	`expanded` Global States
Name From:	author

Role: `heading`

A heading for a section of the page.

This indicates that an object serves as a header. Often, `heading`s will be referenced with the `labelledby` property of the section for which they serve as a header. If headings are organized into a logical outline, the `level` property can be used to indicate the nesting level.

Characteristics:

Characteristic	Value
Parent Roles:	`sectionhead`
Related Concepts:	HTML h1 HTML h2 HTML h3 HTML h4 HTML h5 HTML h6 DTD levelhd
Supported States and Properties:	`level`
Inherited States and Properties:	`expanded` Global States
Accessible Name Required:	True

Role: `list`

Group of non-interactive list items.

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
Parent Roles:	`region`

Child Roles:	`listbox`
	`menu`
	`directory`

Base Concept:	HTML ul

Characteristic	Value
Required Child Elements:	`group` `listitem`
Inherited States and Properties:	`expanded` Global States
Name From:	author

Role: `listitem`

A single item in a list.

Characteristics:

Characteristic	Value
Parent Roles:	`section`
Child Roles:	`treeitem`
Base Concept:	HTML li
Related Concepts:	XForms item
Parent Element:	`list`

Supported States and Properties:	`level`
	`posinset`
	`setsize`

Inherited States and Properties:	`expanded` Global States
Name From:	subtree author
Accessible Name Required:	True

Role: `group`

A section of user interface objects which would *not* be included in a page summary or table of contents by an assistive technology.

Contrast with `region` which is sections of user interface objects that *should* 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. Therefore, proper handling of group by assistive technologies must be determined by the context in which it is provided.

Groups may also be nested. If the author believes a section is significant enough in terms of the entire delivery unit web page then the author should assign the section a role of `region` or a standard landmark role.

Group members that are outside the DOM subtree of the group would need to have explicit relationships assigned to participate in the group using the `owns` property.

Characteristics:

Characteristic	Value
Parent Roles:	`section`

Child Roles:	`select`
	`menubar`
	`toolbar`
	`row`

Related Concepts:	HTML fieldset
Supported States and Properties:	`activedescendant`
Inherited States and Properties:	`expanded` Global States
Name From:	author

Role: `grid`

A `grid` contains cells of tabular data arranged in rows and columns (e.g., a table).

This does 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 grid cells with two dimensional navigation.

Grids **MUST** contain rows with role `row`, which in turn contain cells. Grid cells may be focusable. Grids **MAY** have row and column headers, provided with `rowheader` and `columnheader` roles, which also assist the user agent in supporting navigation. Grid cells **MAY** have contents determined by a calculation.

Grid cells with the `selected` state set can be selected for user interaction, and multiple cells can be selected if the `multiselectable` property of the `grid` is true. Grids may be used for spreadsheets like those in Open Office, Microsoft Office, etc.

Instances of this role **MUST** manage focus of descendants, as described in Managing Focus.

Characteristics:

Characteristic	Value
Parent Roles:	`composite` `section`
Child Roles:	`treegrid`
Base Concept:	HTML table
Required Child Elements:	`row`

Characteristic	**Value**

Supported States and Properties:	`level`
	`multiselectable`
	`readonly`

Inherited States and Properties:	`activedescendant`
	`expanded`
	Global States

Name From:	author
Accessible Name Required:	True

Role: `row`

A row of grid cells.

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

In a `treegrid`, rows **MAY** be expandable, using the `expanded` state to indicate the present status. This is not the case for an ordinary `grid`, in which the `expanded` state is not present.

Characteristics:

Characteristic	**Value**
Parent Roles:	`group`
Base Concept:	HTML tr
Parent Element:	`grid` `treegrid`
Required Child Elements:	`gridcell`
Supported States and Properties:	`level` `selected`

Inherited States and Properties:	`activedescendant`
	`expanded`
	Global States

Name From:	subtree author

Role: `gridcell`

A cell in a grid.

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

Grid cells **SHOULD** explicitly indicate which header cells are relevant to them. They do this by referencing elements with role `rowheader` or `columnheader` using the `describedby` property.

In a treegrid, cells **MAY** be expandable and use the `expanded` state. If the `expanded` property 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 property on a cell is pivot table type behavior.

Characteristics:

Characteristic	Value
Parent Roles:	`section` `widget`
Child Roles:	`rowheader` `columnheader`
Base Concept:	HTML td
Parent Element:	`row`

Supported States and Properties:	`level`
	`readonly`
	`selected`

Inherited States and Properties:	`expanded` Global States
Name From:	subtree author
Accessible Name Required:	True

Role: `rowheader`

A table cell containing header information for a row.

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

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

Characteristics:

Characteristic	Value
Parent Roles:	`gridcell` `sectionhead`
Base Concept:	HTML th with scope=row
Parent Element:	`row`
Supported States and Properties:	`sort`

Inherited States and Properties:	`expanded`
	`level`
	`readonly`

Characteristic	Value
	`selected` Global States
Name From:	subtree author
Accessible Name Required:	True

Role: `columnheader`

A table 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 a structural equivalent to an HTML `th` element with a "column" scope.

> Note: because grid cells are organized into rows, there is not a single container element for the column. The column is the set of `gridcell`s in a particular position within their respective `row` containers.

Characteristics:

Characteristic	Value
Parent Roles:	`gridcell` `sectionhead`
Base Concept:	HTML th with scope=col
Parent Element:	`row`
Supported States and Properties:	`sort`

Inherited States and Properties:	`expanded`
	`level`
	`readonly`
	`selected`
	Global States

Name From:	subtree author
Accessible Name Required:	True

Role: `treegrid`

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

Instances of this role **MUST** manage focus of descendants, as described in

Managing Focus.

Characteristics:

Characteristic	**Value**
Parent Roles:	`grid` `tree`
Required Child Elements:	`row`

Inherited States and Properties:	`activedescendant`
	`expanded`
	`level`
	`multiselectable`
	`readonly`
	Global States

Name From:	author
Accessible Name Required:	True

Role: `description`

Descriptive content for a page element.

A `description` **MUST** be referenced from the element it describes via `describedby`.

Characteristics:

Characteristic	**Value**
Parent Roles:	`section`
Child Roles:	`tooltip`
Inherited States and Properties:	`expanded` Global States

Role: `directory`

A list of references to members of a single group.

Authors **SHOULD** use this role for static tables of contents. This includes tables of contents built with lists, including nested lists. Dynamic tables of contents, however, would be a `tree`.

> Note: directories do not have to contain links. They can have simply unlinked references.

Characteristics:

Characteristic	**Value**
Parent Roles:	`list`
Child Roles:	`tablist`

Characteristic	Value
Related Concepts:	DAISY Guide
Inherited States and Properties:	`expanded` Global States
Name From:	subtree author

Role: `img`

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. Elements with a role of img **MUST** have alternative text or a label associated via the `labelledby` property.

Characteristics:

Characteristic	Value
Parent Roles:	`section`
Related Concepts:	DTB imggroup
Inherited States and Properties:	`expanded` Global States
Name From:	author
Accessible Name Required:	True
Children Presentational:	True

Role: `presentation`

An element whose role is decorative, not meaningful, and does not need to be mapped to the accessibility API.

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.

Example use cases:

- A layout table;
- An `object` in HTML whose content is decorative like a white space image or decorative object;
- An image used for white space;
- A `div` in HTML used to force line breaks before and after its contents.

The user agent **MAY** choose not to present all structural aspects of the element being repurposed. For example, for a table marked as `presentation`, the user agent would remove the `table`, `td`, `th`, `tr`, etc. elements from the accessibility API mapping, while preserving the individual text elements within them. Because the user agent knows to ignore the structural aspects implied in a table, no harm is done by using a table for layout.

User agents **SHOULD** ignore elements with the presentation role for the purpose of determining the containment of items with ARIA roles. For example, in the following code sample:

```
<ul role="group">
  <li role="presentation">
    <a role="presentation>...</a>
  </li>
</ul>
```

the `a` should be considered a child of the `ul` element with the `group` role.

Characteristics:

Characteristic	**Value**
Parent Roles:	`structure`
Inherited States and Properties:	`expanded` Global States

Role: `separator`

A line or bar that separates and distinguishes sections of content.

This is a visual separator between sections of content. For example, separators are found between groups of menu items in a menu.

Characteristics:

Characteristic	**Value**
Parent Roles:	`structure`
Related Concepts:	HTML hr
Inherited States and Properties:	`expanded` Global States
Name From:	author
Children Presentational:	True

Role: `math`

Content that represents a mathematical expression.

This is for sections that represent math, such as images and ASCII art, but are not in a formal mathematical language. Such images **MUST** be labeled by text that can be converted to an accessible format, using the `describedby` property with a reference to a description of the math expression as it would be spoken. This is designed to facilitate conversion to speech. The text description should have no special markup used to control a speech device. Authors **MAY** store image alternative text in the image formats themselves. In these scenarios the alternative text from the image **SHOULD** also be brought into the text of the document and referenced by the `describedby` property.

Characteristics:

Characteristic	Value
Parent Roles:	`section`
Inherited States and Properties:	`expanded` Global States
Name From:	author
Children Presentational:	True

4.4.5. Specialized Regions

These are special types of self-contained aspects of the user interface.

Roles in this section include:

- `application`
- `dialog`
- `alert`
- `alertdialog`
- `marquee`
- `log`
- `status`
- `progressbar`
- `timer`

Role: `application`

A software unit executing a set of tasks for its users.

The intent is to hint to the assistive technology to switch its normal browsing mode functionality to one in which they would for an application. Screen readers have a browse navigation mode where keys, such as up and down arrows, are used to browse the document. This breaks use of these keys by a web application.

To properly set the role of `application`, an author should set the `application` role on an element which encompasses the entirety of the region for which assistive technology browser navigation mode is applied. If it applies to the entire Web page, it should be set on the root node for content, e.g., `body` in HTML or `svg` 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. Much of this navigation would be defined by the application author. However, when reading an email message the content should appear in a region with a `document` role in order to use browsing navigation.

All non-decorative static text or image content inside the application **MUST** be either associated with a form widget or `group` via `labelledby` or `describedby`, or separated out into an element with role of `document` or `article` of its own.

Applications **MUST** have a document title or label. This should be suitable for use as a navigation preview or table-of-contents entry for the page section in question. The label **SHOULD** come from one of the following sources:

- The `title` element in HTML labels the entire document;
- Any `group` element in SVG may contain a non-empty `title` element;
- The element with the `application` role may have a `labelledby` property referencing one or more elements with non-empty text content.

Characteristics:

Characteristic	Value
Parent Roles:	`region`
Related Concepts:	Device Independence Delivery Unit
Inherited States and Properties:	`expanded` Global States
Name From:	author
Accessible Name Required:	True

Role: `dialog`

A dialog is a small application window that sits above the application and is designed to interrupt the current processing of an application in order to prompt the user to enter information or require a response.

Dialog boxes **SHOULD** have a title, which may be provided with a `labelledby` property if other mechanisms are not available. They **MUST** have a focused item, i.e., a descendant element that has the keyboard focus, which is managed by the user agent.

Characteristics:

Characteristic	Value
Parent Roles:	`window`
Child Roles:	`alertdialog`
Inherited States and Properties:	`expanded` Global States
Name From:	author
Accessible Name Required:	True

Role: `alert`

A message with important information.

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 container of the subtree containing the alert message. Alerts are specialized forms of the `status` role, which should be processed as an atomic live region.

Alerts do not require user input and therefore should not receive focus. Since alerts do not receive focus users **SHOULD NOT** be required to close an alert. The user agent **MAY** fire an accessibility alert event, when the alert is created, provided one is specified by the intended accessibility API. If an alert requires focus to close the alert, then an `alertdialog` **SHOULD** be used instead.

Characteristics:

Characteristic	Value
Parent Roles:	`region`
Child Roles:	`alertdialog`
Related Concepts:	XForms alert
Inherited States and Properties:	`expanded` Global States
Name From:	author

Role: `alertdialog`

A separate window (may be simulated) with an alert, where initial focus goes to the window or a widget within it.

Alert dialogs are used to convey messages to alert the user. The alertdialog role goes on the container of the subtree containing the alert message.

Unlike `alert`, `alertdialog` can receive a response from the user, such as to confirm that the user understands the alert being generated. When the alert dialog is displayed, authors **MUST** set focus to an active element within the alert dialog, such as a form edit field or an ok pushbutton. The user agent **MAY** fire an accessibility alert event, when the alert is created, provided one is specified by the intended accessibility API.

Characteristics:

Characteristic	Value
Parent Roles:	`alert` `dialog`
Related Concepts:	XForms alert

Characteristic	Value
Inherited States and Properties:	`expanded` Global States
Name From:	author
Accessible Name Required:	True

Role: `marquee`

A marquee is used to scroll text across the page.

A common usage is a stock ticker. A `marquee` behaves like a live region, with an assumed default `live` property value of "off". An example of a marquee is a stock ticker. A major difference between a marquee and a `log` is how fast it gets updates from timed or real world events.

Characteristics:

Characteristic	Value
Parent Roles:	`section`
Inherited States and Properties:	`expanded` Global States
Accessible Name Required:	True

Role: `log`

A region where new information is added and old information may disappear.

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.

Characteristics:

Characteristic	Value
Parent Roles:	`region`
Inherited States and Properties:	`expanded` Global States
Name From:	author
Accessible Name Required:	True

Role: `status`

Container for processing advisory information to give feedback to the user.

A `status` object must have content within it to provide the actual status information. This object **SHOULD NOT** receive focus.

Status is a form of live region. Its assumed default value for the `live` property is "polite".

If another part of the page controls what appears in the status, the relationship **SHOULD** be made explicit with the `controls` property.

Some cells of a Braille display **MAY** be reserved to render the status.

Characteristics:

Characteristic	Value
Parent Roles:	`composite` `region`
Child Roles:	`timer`

Inherited States and Properties:	`activedescendant`
	`expanded`
	Global States

Role: `progressbar`

Displays the execution status for tasks that take a long time to execute.

This lets the user know that the user's action request has been accepted and that the application continues (or ceases, in the case of a static display) to make progress toward completing the requested action. The author **SHOULD** supply values for `valuenow`, `valuemin`, and `valuemax`, unless the value is indeterminate in which the property should be omitted. These values **SHOULD** be updated 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 `describedby` to point to the status, and set the `busy` state to "true" on the region until it is finished loading.

Characteristics:

Characteristic	Value
Parent Roles:	`widget`
Related Concepts:	`status`

Supported States and Properties:	`valuemax`
	`valuemin`
	`valuenow`
	`valuetext`

Inherited States and Properties:	Global States
Name From:	author
Accessible Name Required:	True
Children Presentational:	True

Role: `timer`

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. However, the timer value is not necessarily machine parsable. The text contents **MUST** be updated at fixed intervals, except when the timer is paused or reaches an end-point.

A timer is a form of live region. The default value of `live` for `timer` is "off".

Characteristics:

Characteristic	**Value**
Parent Roles:	`status`
Inherited States and Properties:	`activedescendant` `expanded` Global States
Name From:	author
Accessible Name Required:	True

4.4.6. Landmark Roles Inherited from the XHTML Role Attribute Module

This section includes roles imported from the *XHTML Role Attribute Module* [*XHTML-ROLES*, Section 4]. These roles are included here in order to make them clearly part of the ARIA Role taxonomy. Role descriptions are taken from the description provided in the XHTML Role Attribute Module.

Roles in this section include:

- `article`
- `banner`
- `complementary`
- `contentinfo`
- `main`
- `navigation`
- `search`

Role: `article`

Represents a section of a page consisting of a composition forming an independent part of a document, page, or site.

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 inner articles represent articles that are in principle related to the contents of the outer 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. Assistive technologies must treat and article like a document in that article must must be processed like an application. Unlike a document, the use of articles allows the user to identify them and follow related articles based on the nesting.

When applying the article to a host language element, ensure that the element and the corresponding end tag wrap the entire article.

Characteristics:

Characteristic	Value
Parent Roles:	`document` `region`
Related Concepts:	HTML 5 article
Inherited States and Properties:	`expanded` Global States
Name From:	author

Role: `banner`

A region that contains the prime heading or internal title of a page.

Most of the content of a banner is site-oriented, rather than being page-specific. Site-oriented content typically includes things such as the logo of the site sponsor, the main heading for the page, and site-specific search tool. Typically this appears at the top of the page spanning the full width.

Characteristics:

Characteristic	Value
Parent Roles:	`region`
Inherited States and Properties:	`expanded` Global States
Name From:	author

Role: `complementary`

Any section of the document that supports but is separable from the main content, but is meaningful on its own even when separated from it.

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 content should be relevant to the main content; if it is completely separable, a more general role should be used instead.

Characteristics:

Characteristic	Value
Parent Roles:	`region`
Inherited States and Properties:	`expanded` Global States
Name From:	author

Role: `contentinfo`

Meta information which applies to the first immediate ancestor whose role is not presentation.

In the context of the page this would apply to a section or the page of which it is the child. For example, footnotes, copyrights, links to privacy statements, etc. would belong here.

Characteristics:

Characteristic	Value
Parent Roles:	`region`
Inherited States and Properties:	`expanded` Global States
Name From:	author

Role: `main`

Main content in a document.

This marks the content that is directly related to or expands upon the central topic of the page.

Characteristics:

Characteristic	Value
Parent Roles:	`region`
Inherited States and Properties:	`expanded` Global States
Name From:	author

Role: `navigation`

A collection of links suitable for use when navigating the document or related documents.

Characteristics:

Characteristic	Value
Parent Roles:	`region`
Inherited States and Properties:	`expanded` Global States
Name From:	author

Role: `search`

The search tool of a web document.

This is typically a form used to submit search requests about the site or to a more general Internet search service.

Characteristics:

Characteristic	Value
Parent Roles:	`region`
Inherited States and Properties:	`expanded` Global States
Name From:	author

4.4.7. Section Roles inherited from XHTML Role Attribute Module

Role: `definition`

A definition of a term or concept.

A role is not provided to specify the term being defined, although host languages may provide such an element; in HTML this is the `dfn` element. The defined term should be included in such an element even when occurring within an element having the `definition` role.

Characteristics:

Characteristic	Value
Parent Roles:	`section`
Inherited States and Properties:	`expanded` Global States
Name From:	author

Role: `note`

The content is parenthetic or ancillary to the main content of the resource.

Characteristics:

Characteristic	**Value**
Parent Roles:	`section`
Inherited States and Properties:	`expanded` Global States
Name From:	author

5. Supported States and Properties

This section is *normative*.

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 treated nearly identically. However, they are maintained conceptually distinct because there is a subtle difference in their meaning. In the definitions for states and properties in this document, states are introduced with the "State" prefix and properties are introduced with the "Property" prefix.

Editorial Note: The Working Group has not yet decided whether to identify states and properties with the literal attribute names (e.g., `aria-busy`) as implemented in host languages, or continue to use short names without the 'aria-' prefix (e.g., `busy`).

5.1. Characteristics of States and Properties

States and properties have the following characteristics:

5.1.1. 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.

5.1.2. 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.

5.1.3. Inherits into Roles

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

5.1.4. Value

Value restrictions for the state or property. These restrictions are expressed in terms of *XML Schema Datatypes* [XSD]. The following XSD Datatypes are used in this specification.

- boolean: a true/false value, expressed in attribute values as "true" or "false".
- decimal: a real number, represented as a decimal.
- integer: a positive or negative integer, expressed as a decimal number.
- string: an arbitrary string value.
- IDREF: a reference to the ID of another element in the same document.
- IDREFS: a whitespace-delimited list of references to the ID of one or more elements in the same document.
- NMTOKEN: a value selected from a set of allowed values, which are defined by an enumeration constraint.
- NMTOKENS: a whitespace-delimited list of one or more values selected from a set of allowed values, which are defined by an enumeration constraint.
- anyURI: a Uniform Resource Identifier Reference (URI).
- QName: an XML Qualified Name.

Values of type boolean, NMTOKEN, and NMTOKENS are further explained by listing the allowed values and their meanings below the table of characteristics. When a value is indicated as the default, the behavior prescribed by this value **MUST** be followed when the state or property is not provided. Some roles also define what behavior to use when certain states or properties, that do not have default values, are not provided.

> Note: in the XHTML module, value restrictions are necessarily expressed in DTD notation, not XSD. DTD notation does not provide the precise value restrictions supported by XSD, and therefore the values in the DTD often have a wider scope of allowed values than what is actually allowed by this specification. Implementers **MUST** be sure to observe the value restrictions defined here and not rely simply on DTD validation.

5.2. Definitions for States and Properties

States and properties are categorized as follows:

1. Widget states
2. Live Regions
3. Drag and Drop
4. Relationships
5. User Interface Properties

Below is an alphabetical list of ARIA states and properties to be used by rich internet application authors. A detailed definition of the taxonomy supporting these ARIA states and properties follows.

States:

`busy`
: Indicates whether a live region is finished updating.

`checked`
: Indicates the value of a binary or ternary (tri-state) widget such as a checkbox or radio button.

`disabled`
: Indicates that the widget is present, but the value cannot be set.

`dropeffect`
: Shows the effect on the target of a drag and drop operation when the dragged object is released.

`expanded`
: Indicates whether an expandable/collapsible group of elements is currently expanded or collapsed.

`grab`
: Shows an object's state in drag and drop.

`hidden`
: Defines whether or not the object is visible to the user.

`invalid`
: Indicates that data the user has input fails rules established by the application.

`pressed`
: Used for toggle buttons to indicate their current pressed state.

`selected`
: Sets whether the user has selected an item or not.

Properties:

`activedescendant`
: Identifies the current active child of a composite widget.

`atomic`
: Indicates if the assistive technology should present all or part of the changed region to the user when the region is updated.

`autocomplete`
: Indicates whether user input completion suggestions are provided.

`channel`
: Specifies that an alternate method of presentation is recommended, possibly but not necessarily in parallel with other live events.

`controls`
: Defines the elements and subtrees whose contents or presence are controlled by the current element.

`describedby`
: Points to an element which describes the object.

`flowto`
: Establishes the recommended reading order of content, overriding the general default to read in document order.

`haspopup`
: Indicates that the element may launch a pop-up window such as a context menu or submenu.

`labelledby`
: Points to the element which labels the current element.

`level`
: The hierarchical level of an element within a structure.

`live`
: Describes the types of updates the user agent, assistive technology, and user can expect from a live region of Web content.

`multiline`
: Indicates whether a text box accepts only a single line, or if it can accept multiline

input.

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

owns
: Defines a parent/child relationship among elements where the DOM hierarchy cannot be used to represent the relationship. The relationship may represent a visual and functional association between elements that is not defined by any DOM parent/child hierarchy.

posinset
: Indicates an item's number or position within the current level of a tree or list.

readonly
: Indicates that the widget is not editable.

relevant
: Indicates the nature of change within a live region.

required
: Indicates that user input is required on the widget before a form may be submitted.

setsize
: Refers to the number of items in the current level of a list or tree.

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

valuemax
: Maximum allowed value for a range type of widget.

valuemin
: Minimum allowed value for a range type of widget.

valuenow
: The current value of a widget.

valuetext
: The human readable text equivalent of valuenow for a widget.

5.2.1. Widget states and properties

This section contains states specific to common user interface elements found on GUI systems or in rich Internet applications which receive user input and process user actions. These states are used to support the user input and user interface roles. Widget states and properties might be mapped by a user agent to platform accessibility API states, for access by an assistive technology, or they might be accessed directly from the DOM. Changes in states **MUST** result in a notification to an assistive technology either through DOM attribute change events or platform accessibility API events.

States and properties in this section include:

- `autocomplete`
- `checked`
- `disabled`
- `expanded`
- `haspopup`
- `invalid`
- `level`
- `multiline`
- `multiselectable`

- `pressed`
- `readonly`
- `required`
- `selected`
- `valuemax`
- `valuemin`
- `valuenow`
- `valuetext`

Property: `autocomplete`

Indicates whether user input completion suggestions are provided.

For a textbox with autocomplete="inline" or autocomplete="both", the completion text should be selected and come after the caret. The `haspopup` property can be used in conjunction with this to indicate that a popup containing choices appears, notwithstanding the fact that it is a simple text box.

For an element which already has a drop-down (i.e., a `combobox`), it is assumed that the dropdown behavior is still present. This means that if `autocomplete` is true, `haspopup` should also be true on a combobox.

Characteristics:

Characteristic	Value
Related Concepts:	XForms selection attribute in select
Used in Roles:	`textbox`
Value:	NMTOKEN

Values:

Value	Description
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, but the edit box retains focus.
both:	A list of choices appears and the currently selcted suggestion also appears inline.
none:	Only values from the value list can be selected.

State: `checked`

Indicates the value of a binary or ternary (tri-state) widget such as a checkbox or radio button.

The action when a mixed button is activated is covered in *WAI-ARIA Best Practices* [*ARIA-PRACTICES*]

Characteristics:

Characteristic	Value
Used in Roles:	`option`
Inherits into Roles:	`treeitem`
Value:	NMTOKEN

Values:

Value	Description
true:	The current item is checked.
false:	The role supports being checked but is not currently checked.
mixed:	Indicates a mixed mode value for a tri-state checkbox. This is not supported on `radio` or `menuitemradio` or any element that inherits from these in the taxonomy; user agents **MUST** treat a mixed value as equivalent to "false" on those roles. Direct user action on the object having the checked state cannot automatically put it into the mixed state.
undefined:	The object does not support being checked.

State: `disabled`

Indicates that the widget is present, but the value cannot be set.

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. There **SHOULD** be a change of appearance to indicate that the item has been disabled (grayed out, etc.).

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

Characteristics:

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

Values:

Value	Description
true:	The widget and all focusable descendants are disabled and its value cannot be changed by the user.
false:	The widget is enabled.

State: `expanded`

Indicates whether an expandable/collapsible group of elements is currently expanded or collapsed.

For example, this indicates whether a portion of a tree is expanded or collapsed.

Characteristics:

Characteristic	**Value**
Related Concepts:	Tapered prompts in voice browsing. Switch in *SMIL* [*SMIL*].
Used in Roles:	`structure` `window`

Inherits into Roles:	`alert`
	`alertdialog`
	`application`
	`article`
	`banner`
	`columnheader`
	`combobox`
	`complementary`
	`contentinfo`
	`definition`
	`description`
	`dialog`
	`directory`
	`document`
	`grid`
	`gridcell`
	`group`
	`heading`
	`img`
	`list`
	`listbox`
	`listitem`
	`log`
	`main`
	`marquee`
	`math`
	`menu`
	`menubar`
	`navigation`
	`note`
	`presentation`
	`radiogroup`
	`region`
	`row`
	`rowheader`
	`search`
	`section`
	`sectionhead`
	`select`
	`separator`
	`status`
	`tab`

Characteristic	Value
	tablist tabpanel timer toolbar tooltip tree treegrid treeitem
Value:	NMTOKEN

Values:

Value	Description
true:	The group is expanded.
false:	The group is collapsed.
undefined:	The group is neither expandable nor collapsible; all its child elements are shown or there are no child elements.

Property: `haspopup`

Indicates that the element may launch a pop-up window such as a context menu or submenu.

This means that activation renders conditional content. Note that ordinary tooltips are not considered popups in this context. The drop-down options in a `combobox` are also not popups in this sense.

A popup is generally presented visually as a bordered group of items that appears to be on top of the main page content. If possible, the entire popup should be visible when it opens (not partially offscreen).

Characteristics:

Characteristic	Value
Related Concepts:	This is a sub property of `controls` *User Agent Accessibility Guidelines* [*UAAG*] conditional content
Used in Roles:	All elements of the base markup
Value:	boolean

Values:

Value	Description
true:	Indicates the object has a popup, either as a descendant or pointed to by `owns`.
false:	The object has no popup.

State: `invalid`

Indicates that data the user has input fails rules established by the application.

If the value is computed to be invalid or out-of-range, this value should be set to true. User agents **SHOULD** inform the user of the error. Applications **SHOULD** provide suggestions for correction where they are known. User agents **MAY** refuse to submit the form as long as there is a widget for which `invalid` is true.

When the user attempts submit data involving a field for which `required` is true, the application **MAY** use the invalid property to signal there is an error. However, the invalid property should not be set on required widgets simply because the user has not yet input data but has not not attempted to submit it.

For future expansion, the `invalid` state is an enumerated type. Any value not recognized in the list of allowed values **MUST** be treated as if the value "true" had been provided. The default value, however, is still "false". If the state is not present, its value is "false", or its value is the zero length string (it is present with a blank value), this default applies.

The `invalid` property applies only to the element on which it is applied. The state of being invalid does not propogate either to descendant elements nor to ancestor elements.

Characteristics:

Characteristic	Value
Related Concepts:	*XForms* [*XForms*] 'invalid' event http://www.w3.org/TR/2006/REC-xforms-20060314/slice4.html#evt-revalidate. Note: This state is true if a form field is required but empty. However, XForms's valid property would be set to false.
Used in Roles:	All elements of the base markup
Value:	NMTOKEN

Values:

Value	Description
true:	The value entered by the user has failed automated validation.
false:	There are no detected errors in the value.
spelling:	A spelling error was detected.
grammar:	A grammatical error was detected.

Property: `level`

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, and to other structural items that may appear inside a container or participate in an ownership hierarchy. Levels **MUST** be 1 or greater.

Levels should increase with depth increases relative to the level of the parent. Level information should be maintained by the author.

This property **MUST** be applied to leaf nodes (elements that would receive focus), not to the parent grouping element, even when all siblings are at the same level. This means that multiple elements in a set may have the same value for this property. Although it would be less repetitive to provide a single value on the container, it is not always possible for authors to do so. Restricting this to leaf nodes ensures that there is a single way for assistive technology to use the property.

If the DOM ancestry accurately represents the level, the user agent can calculate the level of an item from the document structure. This property can be used to provide an explicit indication of the level when that is not possible to calculate from the document structure or the `owns` property. User agent automatic support for automatic calculation of level may vary; authors should test with user agents and assistive technologies to determine whether this property is needed. If the author intends for the user agent to calculate the level, they **MUST** omit this property.

Characteristics:

Characteristic	Value
Used in Roles:	`grid` `gridcell` `heading` `row` `listitem`
Inherits into Roles:	`columnheader` `rowheader` `treegrid` `treeitem`
Value:	integer

Property: `multiline`

Indicates whether a text box accepts only a single line, or if it can accept multiline input.

There is very little difference in ARIA between single-line and multi-line text boxes, as both allow arbitrary text input. The main reason to indicate this is to warn of different behaviors of the `enter` key. In a multi-line text box, the `enter` key adds a new line; in a single-line text box, it does not and may activate a function outside the text box such as submitting the form.

Characteristics:

Characteristic	Value

Characteristic	Value
Used in Roles:	`textbox`
Value:	boolean

Values:

Value	Description
true:	This is a multi-line text box.
false:	This is a single-line text box.

Property: `multiselectable`

Indicates that the user may select more than one item from the current selectable descendants.

Lists, trees, and grids may allow users to select more than one item at a time.

Descendants that are selected are indicated with the `selected` state set to "true". Descendants that are selectable but not selected are indicated with the `selected` state set to "false". Descendants that are not selectable should not set the `selected` state.

Characteristics:

Characteristic	Value

Used in Roles:	`grid`
	`listbox`
	`tree`

Inherits into Roles:	`treegrid`
Value:	boolean

Values:

Value	Description
true:	More than one item in the widget may be selected at a time.
false:	Only one item can be selected.

State: `pressed`

Used for toggle buttons to indicate their current pressed state.

Toggle buttons require a full press and release cycle to toggle their value. Activating it once changes the pressed state to "true", and activating it another time changes the pressed state back to "false". A value of "mixed" means that the states of more than one item controlled by the button do not all share the same value; the action when a mixed button is activtate is covered in *WAI-ARIA Best Practices* [*ARIA-PRACTICES*]. If the state is not present, the value defaults to "undefined" meaning the button is not a toggle button.

The pressed state is similar but not identical to the `checked` state. Operating systems support `pressed` on buttons and `checked` on checkboxes.

Characteristics:

Characteristic	Value
Used in Roles:	`button`
Value:	NMTOKEN

Values:

Value	Description
true:	The button is pressed.
false:	The button is not pressed.
mixed:	The elements on the page that are affected by the button do not all share the same value. Direct user action on the object having the pressed state cannot automatically put it into the mixed state.
undefined:	The button is not a toggle button and activating it does not change this state.

Property: `readonly`

Indicates that the widget is not editable.

This means the user can read but not set the value of the widget. Readonly objects are relevant to the user and applications **MUST NOT** restrict navigation to focusable descendants. Other actions such as copying the value of the widget are also supported. This is in contrast to `disabled` objects for which applications might choose not to allow users to navigate to descendants.

Examples include:

- A form element which represents a constant.
- Row or column headers in a spreadsheet.
- The result of a calculation such as a shopping cart total.

Characteristics:

Characteristic	Value
Related Concepts:	*XForms* [*XForms*] Readonly
Used in Roles:	`grid` `gridcell` `textbox`
Inherits into Roles:	`columnheader` `rowheader` `treegrid`
Value:	boolean

Values:

Value	Description
true:	The user cannot change the value of the widget.
false:	The user can set the value of the widget.

Property: `required`

Indicates that user input is required on the widget before a form may be submitted.

For example, if a user must fill in an address field, then `required` is set to "true".

> Note: the fact that the element is required is often visually presented (such as a sign or symbol after the widget). Using the required attribute makes it much easier for user agents to pass on this important information to the user.

The `required` property applies only to the element on which it is applied. The property of being required does not propogate either to descendant elements nor to ancestor elements.

Characteristics:

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

Values:

Value	Description
true:	Users must provide input on a widget before a form is submitted.
false:	User input is not necessary to submit the form.

State: `selected`

Sets whether the user has selected an item or not.

Selecting an element indicates that it is chosen for an action, and most likely has focus. However, this does not imply anything about other states.

This property is used in two cases:

1. Single selection containers where the currently focused item is not selectable. In this case the selection normally follows the focus, and is managed by the user agent. In this case the author should only use `selected` when the focused item is not in fact selected. The only useful value is "false" because otherwise the currently focused item is considered to be selected.

2. Multiselectable containers. Any selectable descendant of a container in which the `multiselectable` property is "true" must actively specify a value of either "false" or "true" for the `selected` state.

Characteristics:

Characteristic	Value
Used in Roles:	`gridcell` `option` `row`
Inherits into Roles:	`columnheader` `rowheader` `treeitem`
Value:	NMTOKEN

Values:

Value	Description
true:	A selectable element is actually selected.
false:	The element is not selected but is selectable.
undefined:	The element is not selectable.

Property: `valuemax`

Maximum allowed value for a range type of widget.

A range widget may start with a given value, which can be increased until a maximum value, defined by this property, is reached.

Declaring the `valuemax` will allow for alternate device to calibrate an arrow up effect, validate, or simply let the user know the size of the range on offer. If the `valuenow` has a known maximum and minimum, the author **SHOULD** provide properties for `valuemax` and `valuemin`.

Characteristics:

Characteristic	Value
Related Concepts:	*XForms* [*XForms*] range
Used in Roles:	`progressbar` `range`
Inherits into Roles:	`slider` `spinbutton`
Value:	decimal

Property: `valuemin`

Minimum allowed value for a range type of 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 `valuemin` allows for alternate device to calibrate an arrow up effect, validate, or simply let the user know the size of the range on offer. If the `valuenow` has a known maximum and minimum, the author **SHOULD** provide properties for `valuemax` and `valuemin`.

Characteristics:

Characteristic	Value
Related Concepts:	*XForms* [*XForms*] range
Used in Roles:	`progressbar` `range`
Inherits into Roles:	`slider` `spinbutton`
Value:	decimal

Property: `valuenow`

The current value of a widget.

Used, for example, for a range widget such as a slider or progress bar, and for a date.

If the value is not known (as often occurs with progress bars) then the `valuenow` attribute should not be set at all. If the valuenow attribute is absent, no information is implied about the current value. If the `valuenow` has a known maximum and minimum, the author **SHOULD** provide properties for `valuemax` and `valuemin`.

The type of valuenow is a number. If the value type of the widget is not a number, provide an index value for this field (i.e., a numeric value that represents the position of the present value within the set of possible values), plus a proper value with the `valuetext` property.

Characteristics:

Characteristic	Value
Related Concepts:	*XForms* [*XForms*] range, start
Used in Roles:	`progressbar` `range`
Inherits into Roles:	`slider` `spinbutton`
Value:	decimal

Property: `valuetext`

The human readable text equivalent of `valuenow` for a widget.

Used, for example, for a range widget such as a slider or progress bar.

In order for the `valuetext` property to be set, the `valuenow` property **MUST** also be set (as often occurs with progress bars). The `valuetext` should be set by authors only when the rendered slider value cannot be completely be represented in the form of a number. For example, a slider may has rendered values of {"small", "medium", "large", "extra large"}. In this instance the values of `valuenow` range from 0 through 3, which indicate the position of each value in the value space, but the `valuetext` would be one of the strings: "small", "medium", "large", and "extra large". If the `valuetext` property is absent, the assistive technology will rely solely on the `valuenow` property for the current value.

Characteristics:

Characteristic	**Value**
Related Concepts:	*XForms* [*XForms*] range, start
Used in Roles:	`progressbar` `range`
Inherits into Roles:	`slider` `spinbutton`
Value:	string

5.2.2. Live Regions

This section contains properties specific to live regions in rich Internet applications. These properties may be applied to any element. The purpose of these properties is to indicate that changes to the section on which they are applied may occur without it having focus, and to provide the assistive technology information on how to process live updates in this section of the page. Some roles specify a default value for the `live` property specific to that role.

Examples of live regions include:

- A section of updated basketball statistics.
- A region that updates in response to a user's control of a Web page (such as requesting a photograph).

States and properties in this section include:

- `atomic`
- `busy`
- `channel`
- `live`
- `relevant`

Property: `atomic`

Indicates if the assistive technology should present all or part of the changed region to the user when the region is updated.

Both accessibility APIs and the *Document Object Model* [*DOM*] provide events to allow the assistive technology to determine changed areas of the document.

When a node changes, the AT **SHOULD** look at the changed element and then traverse the ancestors to find the first element with `atomic` set, and apply the appropriate behavior for the cases below.

1. If none of the ancestors have explicitly set `atomic`, the default is that `atomic` is "false", and the AT only needs to present the changed node to the user.
2. If `atomic` is explicitly set to "false", then the AT can stop searching up the ancestor chain, and should present only the changed node to the user.
3. If `atomic` is explicitly set to "true", then the AT should present the entire subtree of the element on which `atomic` was set.

When `atomic` is true, the AT **MAY** choose to combine several changes and present the entire changed region at once.

If the region contains only a single data field and a label, then `labelledby` can be used instead of `atomic`, to ensure the entire region is spoken during a change.

Characteristics:

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

Values:

Value	**Description**
true:	Tthe assistive technology should present the entire region as a whole.
false:	A change within the region may be processed by the assistive technology on its own.

State: `busy`

Indicates whether a live region is finished updating.

The default is that `busy` is "false". For example, if authors know that multiple parts of the same live region need to be loaded, they can set `busy` to "true" when the first part is loaded, and then set `busy` to "false" or remove the attribute when the last part is loaded. If there is an error updating the live region, set the `invalid` property to "true".

Characteristics:

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

Values:

Value	Description
true:	The live region is still being updated.
false:	There are no more expected updates for that live region.

Property: `channel`

Specifies that an alternate method of presentation is recommended, possibly but not necessarily in parallel with other live events.

The default channel is "main".

The mapping of hardware channels (speech synthesizer, braille, etc.) to the "main" and "notify" channels is implementation/configuration dependent.

If multiple hardware channels are available, the AT **SHOULD** allow users to map the "main" and "notify" channels as they wish.

If there is only one hardware channel available, the AT **SHOULD** render both channels on the same hardware channel. The "notify" channel is higher priority than the "main" channel for live regions of the same politeness (which is a value of the `live` property). If the events from the two channels are of differing politeness levels, the channel with the higher priority event **SHOULD** have higher priority than the other channel. Events from one channel **MUST NOT** interrupt or clear out events on another channel.

Characteristics:

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

Values:

Value	Description
main:	The default channel on which to send live region events.
notify:	A channel with higher priority than the "main" channel on which to present live region changes to the user.

Property: `live`

Describes the types of updates the user agent, assistive technology, and user can expect from a live region of Web content.

It is essential to describe the types of updates the user agent and user can expect from a live region of Web content. The values of this state are expressed in terms of "politeness" levels. "Polite" regions notify of updates but do not interrupt users, and updates take low priority. An appropriate use of more assertive content would be to notify users of a site that the connection is going down in 5 minutes, since the importance of the message is greater than the problem caused by the interruption.

When the property is not set on an object that needs to send updates, the politeness level is inherited from the value of the nearest ancestor that sets the `live` property.

Characteristics:

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

Values:

Value	**Description**
off:	The region is not currently live.
polite:	(Background change) This is normal operation and **SHOULD** be the default behavior for live regions. It is not necessary to respond until user completes their current activity.
assertive:	This information has a higher priority than normal but does not necessarily interrupt immediately.
rude:	This information has the highest priority and should typically result in an interrupt to the user. This may disorientate users causing them not to continue in their current task.

Property: `relevant`

Indicates the nature of change within a live region.

The property 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 relevant should be set to "all". When the `relevant` property is not provided, the default is to assume that text changes and node additions are relevant, and that node removals are not relevant.

`relevant` is an optional property of live regions within a document. It does not restrict how an assistive technology processes attributes. This is a hint to the AT, but the AT is still not required to present changes of all the relevant types.

Both accessibility APIs and the *Document Object Model* [DOM] provide events to allow the assistive technology to determine changed areas of the document.

When this value is not set, the value inherits from an object's nearest ancestor. It is not additive, meaning the set of values provided and omitted on an object completely override any inheritance of values.

Characteristics:

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

Values:

Value	**Description**
additions:	Nodes are added to the DOM within the region.
removals:	Nodes are removed from the DOM.
text:	Text is added or removed from the DOM.
all:	Equivalent to the value "additions removals text".
additions text:	Nodes are added or text is added to or removed from the DOM.

5.2.3. Drag and Drop

This section defines properties which must be applied by an author to indicate the state of objects which may be "grabbed" for a drag operation as well as the state of drop targets once a drag operation has been started. This information is designed to facilitate a drag and drop operation by identifying "draggable" objects and drop target. This information should either be rendered visually or provided to the user by an assistive technology through an alternative modality.

For more information about using drag and drop, see *Drag-and-Drop Support in the ARIA Best Practices* ([*ARIA-PRACTICES*], Section 7).

States and properties in this section include:

- `dropeffect`
- `grab`

State: `dropeffect`

Shows the effect on the target of a drag and drop operation when the dragged object is released.

More than one drop effect may be supported for a given element. Therefore, the value of this state is a space-delimited set of tokens indicating the possible effects, or "none" if there is no supported operation. This state also allows authors to use a style sheet to provide a visual indication of the target (e.g., highlight it) during drag operations. If only one type of operation is supported, it can be set at page load.

Characteristics:

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

Characteristic	Value
Value:	NMTOKENS

Values:

Value	Description
copy:	A duplicate of the source object will be dropped into the target.
move:	The source object will be removed from its original location and dropped into the target.
reference:	A reference or short cut to the dragged object will be created in the target object.
execute:	A function supported by the drop target is executed, using the drag source as an input.
popup:	The author **MUST** provide a popup menu or dialog to allow the user to choose one of the drag operations (copy, move, reference) and any other drag functionality, such as drag cancel.
none:	No operation can be performed; effectively cancels the drag operation if an attempt is made to drop on this object.

State: `grab`

Shows an object's state in drag and drop.

When it is set to "true" it has been selected for dragging, "supported" indicates that the object is grabbable, but is not currently grabbed, "false" indicates the object is not grabable (default).

Characteristics:

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

Values:

Value	Description
true:	Indicates that the element has been "grabbed" for dragging.
supported:	Indicates that the element supports being dragged.
false:	Indicates that the element does not support being dragged.

5.2.4. Relationships

This section defines relationships or associations between elements which cannot be readily determined from the document structure.

States and properties in this section include:

- `activedescendant`
- `controls`

- `describedby`
- `flowto`
- `labelledby`
- `owns`
- `posinset`
- `setsize`

Property: `activedescendant`

Identifies the current active child 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, spreadsheets.

Authors **SHOULD** ensure that the object targeted by the `activedescendant` property is either a descendant of the container in the DOM, or is a logical descendant as indicated by the `owns` property. The user agent is not expected to check that the activedescendant is an actual descendant of the container. Authors **SHOULD** ensure that the currently activedescendant is currently visible in the viewport (not scrolled off). Authors **SHOULD** capture changes to the activedescendant property, which can occur if the AT sets focus directly.

Characteristics:

Characteristic	Value
Related Concepts:	*SVG* [*SVG*] and *DOM* [*DOM*] active
Used in Roles:	`composite` `group`

Inherits into Roles:	`combobox`
	`grid`
	`listbox`
	`menu`
	`menubar`
	`radiogroup`
	`row`
	`select`
	`spinbutton`
	`status`
	`tablist`
	`timer`
	`toolbar`
	`tree`
	`treegrid`

Value:	IDREF

Property: `controls`

Defines the elements and subtrees whose contents or presence are controlled by the current element.

For example:

- A tree view table of contents may control the contents of a neighboring help contents document pane.
- A group of checkboxes may control what commodity prices are tracked live in a table or graph.

Characteristics:

Characteristic	Value
Related Concepts:	*XML Events* [*XML events*] object hyperlink target in *HTML* [*HTML*]
Used in Roles:	All elements of the base markup
Value:	IDREFS

Property: `describedby`

Points to an element which describes the object.

This is very similar to labeling an object with `labelledby`. A label should provide the user with the essence of the what the object does, whereas a description is intended to provide additional information that some users might need.

Characteristics:

Characteristic	Value
Related Concepts:	Related concepts: Hint or Help in *XForms* [*XForms*] Label in XForms Label in *HTML* [*HTML*] online help HTML table cell headers HTML label element, and HTML table cell headers are de facto describedby values.
Used in Roles:	All elements of the base markup
Value:	IDREFS

Property: `flowto`

Establishes the recommended reading order of content, overriding the general default to read in document order.

When flowto has a single IDREF, it instructs assistive technology to skip normal document reading order and go to the targeted object. Flowto in subsequent elements would follow a process similar to *next focus in XHTML2* ([XHTML], Section 13). However, when flowto is provided with multiple IDs, then they should be processed as path choices by the AT, such as in a model based authoring tool. This means that the user **SHOULD** be given the option of navigating to any of the elements targeted. The name of the path can be determined by the name of the target element of the `flowto`. Accessibility APIs can provide named path relationships.

Characteristics:

Characteristic	**Value**
Related Concepts:	*XHTML 2* [*XHTML2*] :nextfocus :prevfocus
Used in Roles:	All elements of the base markup
Value:	IDREFS

Property: `labelledby`

Points to the element which labels the current element.

This is very similar to describing an object with `describedby`. A label should provide the user with the essence of the what the object does, whereas a description is intended to provide additional information that some users might need.

> Note: the expected spelling of this property in U.S. English would be "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* [*XForms*] and *HTML* [*HTML*].
Used in Roles:	All elements of the base markup
Value:	IDREFS

Property: `owns`

Defines a parent/child relationship among elements where the DOM hierarchy cannot be used to represent the relationship. The relationship may represent a visual and functional association between elements that is not defined by any DOM parent/child hierarchy.

The value of the `owns` attribute is a space-separated list of IDREFs; that is, unique identifiers that reference one or more elements in the document. Each of the latter elements have an 'id' attribute that matches one of the values in the list of IDREFs.

The reason for adding `owns` is to expose parent/child relationships to an AT that would otherwise be missed. In that sense `owns` is more important than DOM-parentage. It publicizes a relationship that is otherwise difficult or impossible to infer from the DOM.

Authors **MUST NOT** use `owns` as a replacement for the DOM hierarchy. That is, if the relationship is captured by the DOM, then do not use `owns`.

Characteristics:

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

Property: `posinset`

Indicates an item's number or position within the current level of a tree or list.

For example, if this element is the third item in a group then posinset is equal to three. The range of values is 1 to the size of the set.

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. User agent support for automatic calculation of position and setsize may vary. Authors **SHOULD** test with user agents and assistive technologies to determine whether this property is needed.

Posinset **SHOULD** be used together with `setsize`.

Characteristics:

Characteristic	Value
Used in Roles:	`listitem`
Inherits into Roles:	`treeitem`
Value:	integer

Property: `setsize`

Refers to the number of items in the current level of a list or tree.

For example, if this element is in a group of six items at the same level then setsize is equal to six. Setsize must be >= 1.

This property is marked on the members of a set, not the container element that collects the members of the set. To orient a user to a particular element by saying it is "item N out of M", the client software would use N equal to the `posinset` property on the element and M equal to the `setsize` property on that particular element.

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. User agent support for automatic calculation of position and setsize may vary. Authors **SHOULD** test with user agents and assistive technologies to determine whether this property is needed.

Setsize **SHOULD** be used together with `posinset`.

Characteristics:

Characteristic	Value
Used in Roles:	`listitem`
Inherits into Roles:	`treeitem`
Value:	integer

5.2.5. User interface properties

This section defines properties that affect or describe the rendering of the user interface.

States and properties in this section include:

- `hidden`
- `sort`

State: `hidden`

Defines whether or not the object is visible to the user.

For example, if a menu is only visible after some user action, the `hidden` property should be set to "true" until the menu is presented, at which time the `hidden` property would be removed, indicating that the menu is visible.

It is recommended that authors key visibility of objects off this attribute, rather than change visibility and separately have to remember to update this property. CSS 2 provides a way to *select on attribute values* ([*CSS*], Section 5.8.1). The following pair of CSS declarations make content visible unless the `hidden` property is true; scripts need only update the value of this property to change visibility:

```
[aria-hidden=true] {visibility: hidden;}
```

Note that this CSS example, while technically correct, will not work in some browsers at the time of this writing. It may be necessary to set the style using script.

Characteristics:

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

Values:

Value	Description
true:	Indicates that this section of the document and its children are hidden from the rendered view.
false:	Indicates that this section of the document is rendered.

Property: `sort`

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

This property should be applied only to table or grid headers. If the property is not provided, there is no defined sort order.

Characteristics:

Characteristic	Value
Used in Roles:	`columnheader` `rowheader`
Value:	NMTOKEN

Values:

Value	Description
ascending:	Items are sorted in ascending order by this column.
descending:	Items are sorted in descending order by this column.
none:	There is no special sort applied to the column.
other:	A sort algorithm other than ascending or descending has been applied.

6. Implementation in Host Languages

This section is *normative*.

6.1. Introduction

This section is *informative*.

The roles, states, and properties defined in this specification do not form a whole web language or format. They are intended to be used in the context of a profile based on some host language or other. This section discusses what host languages should do to implement WAI-ARIA, that is to say assure that the markup specified here will integrate smoothly and effectively with their markup in instances of their format.

Although markup languages superficially look alike, they do not all share much by way of language-definition infrastructure. To accomodate differences in language-building approaches, the requirements here have been set out at two levels, general, and modualrization-specific. While allowing for differences in how the specifications are written, we believe we have maintained consistency in how the WAI-ARIA information looks to authors and how it is manipulated in the DOM by scripts.

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 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 name of the state or property.

6.2. General requirements on implementing host languages

6.2.1. Role attribute

An implementing host language **MUST** provide an attribute with the following characteristics:

- the attribute name of this attribute **MUST** be "role";
- the attribute value string for this attribute **MUST** allow a space-separated sequence of whitespace-free substrings;
- the appearance of the name literal of any concrete ARIA role (see section 7.3.2) as one of these substrings **MUST NOT** in and of itself make the attribute value illegal in the host-language syntax;
- the appearance of the name literal of any role defined in the *Role Attribute Module* [*XHTML-ROLES*] as one of these substrings **MUST NOT** in and of itself make the attribute value illegal in the host-language syntax;
- the host language **MUST** provide that where the name literal of a concrete ARIA role appears as one of the substrings in the space-separated list in 'role' attribute value, that that value **MUST** be processed in accordance with this specification;
- the host language **SHOULD** provide that where the name literal of any role defined in the Role Attribute Module appears as one of the substrings in the space-separated list in 'role' attribute value, that that value **MUST** be processed in accordance with that specification.

6.2.2. State and property attributes

An implementing host language **MUST** allow attributes as follows:

- the attribute name of one of these attributes is the name of any state or property identified in Section 5 above, prefixed by "aria-", such as "aria-busy", "aria-selected", "aria-activedescendant", "aria-valuetext";
- the host language syntax **MUST NOT** prevent the attribute from appearing anywhere that it is applicable as specified in this specification;
- the host language **MUST** provide that where these attributes appear in a document instance of their format, these attributes are to be processed as provided in this specification.

Editorial Note: Following the *Namespaces Recommendation* [*XML-NAMES*], the namespace name for these attributes "has no value". The names of these attributes do not have a prefix set off 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.

6.2.3. Focus navigation

An implementing host language **MUST** provide support for

- the author to make arbitrary elements focusable;
- the user to have device-independent capability and in particular keyboard capability to navigate the focus to all focusable elements.

Editorial note: do we have to move the user capability to navigate the focus to a User Agent requirement? What is the role of the host language in this?

6.3. Implementation using the Modularization Recommendation

Host languages can be constructed using the methods of the W3C Recommendation Modularization in XHTML [@@ref2m12n].

Host languages that are constructed following this Recommendation **MUST** implement the abstract module comprising the attributes defined in section 5. above using a module implementation provided in this specification.

Appendix 8.1.2 below provides an implementation of this module as a DTD module.

Appendix 8.1.3 below is a sample DTD integrating a language profile that implements WAI-ARIA by including the DTD module. Note how the applicability of the tabindex attribute has been extended to satisfy 6.2.3 above.

Editorial Note: The URIs to be used to identify the module are still under discussion.

7. Quality Assurance

This section sets out normative conditions for conforming document instances and user agent processors, along with informative remarks about the role of other processors such as authoring tools and assistive technologies.

7.1. Document Conformance

This section is *normative*.

7.1.1. Host language conformance

A document conforming to any Host+ARIA profile **MUST** conform to the specification of the host language, including any provisions in that specification implementing WAI-ARIA in that host language.

7.1.2. Information Pattern Conformance

The applicable ARIA role and all aria-* attributes applied in a conforming document **MUST** conform to the role, property and state definitions set out in sections 4 and 5 above. The applicable ARIA role is extracted from the document markup as described in section 7.3.2 "Applicable ARIA role" below.

7.2. Authoring practices

This section is *informative*.

7.2.1. Authoring tools

W3C publishes *Authoring Tool Accessibility Guidelines* [*ATAG1*]. These documents apply to web-content producing tools including those used for producing ARIA-using content.

Many of the requirements in the definitions of ARIA roles, states and properties can be checked mechanically, although not all. Authoring tools can contribute to the successful use of ARIA markup by offering their users conformance checking modes or operations that screen for the satisfaction of 7.1.1 and 7.1.2 above.

Until such time as a body of good, common practice has been established and confirmed by widespread accessible use, authoring tools would be well advised to pattern their prompting and coaching of authors after the practices in the *Best Practices Guide* [*ARIA-PRACTICES*].

7.2.2. Testing practices and tools

The accessibility of interactive content cannot be confirmed by static checks alone. Developers of interactive content should test for (a) device independent access to the operations of the widgets and applications, and (b) display and API visibility into all states and changes of the widgets and applications during user interaction.

More detailed and current suggestions about authoring and testing tools and methods may be found in the latest version of the *Best Practices Guide* [*ARIA-PRACTICES*].

7.3. User Agent Conformance

This section is *normative*.

7.3.1. 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. This includes but is not limited to the construction of a DOM from a text string.

In applying this rule, however, scripted changes in the DOM of the document and style rules which change the presentation of the content based on selectors sensitive to ARIA markup **MUST** be considered as part of normal operation. On the other hand, the mapping to accessibility APIs **MUST** be considered, in applying this rule, to be above and beyond normal operation. The ARIA processing **MAY** alter the mapping of the host language features into an accessibility API. But the mapping to the API **MUST NOT** alter the DOM of the document.

7.3.2. Applicable ARIA role

7.3.2.1. Requirement is functional

As discussed above in section 6.@@ , an implementing host language will provide a "role" attribute with a value which may be viewed as a space-separated sequence of tokens. The User Agent **MUST** determine zero or one *applicable ARIA role* by computations that achieve the same result as the method described in this section.

7.3.2.2. Overview

The applicable ARIA role is the concrete ARIA role whose name is matched by the first token in the sequence of tokens in the "role" attribute value which matches, on case-sensitive comparison, the name of any concrete ARIA role.

A *concrete ARIA role* is any of the role types described in section 4 above *except* (1) abstract roles, that is roles for which isAbstract is True and (2) the roles imported from the Role Module, that is those that are introduced in sections 4.4.6 and 4.4.7.

7.3.2.3. Step by step

The following steps will correctly identify the applicable ARIA role.

1. The rules of the host language are used to detect that an element has an attribute with attribute name of "role" and to identify the attribute value string for that attribute.
2. The attribute value string for that attribute is broken into a sequence of whitespace-free substrings by separating on whitespace.
3. The substrings are compared in a case-sensitive comparison with all the names of concrete ARIA roles as defined above.
4. The first such substring in textual order that matches the name of a concrete ARIA role is the name of the applicable ARIA role.

7.3.2.4. Zero or one

This sub-paragraph is informative.

As a result, a host language element will have one and only one applicable ARIA role if it has a role attribute and at least one of the tokens in the value of this role attribute matches the name of a concrete ARIA role. It will have zero applicable ARIA roles if it does not have a role attribute, or if the role attribute contains no tokens matching the name of a concrete ARIA role.

7.3.2.5. Explicit ARIA role rules, in accessibility API only

The applicable ARIA role, if there is one, **MUST** be the role value which is mapped to the value of a role property in any accessibility API which accepts only one role value. User agents **MUST** use an explicit applicable ARIA role as overriding any implicit role inferred from the host language markup in performing this mapping. Note that, in conformance with section 7.3.1 above, this overriding does not result in any changes in the DOM, only in the accessibility API representation of the document.

7.3.3. All ARIA in DOM

A conforming User Agent which implements a Document Object Model per the W3C Recommendations **MUST** include the entire "role" attribute value in the DOM and all ARIA states and properties in the document instance in the corresponding DOM.

Editorial Note:

- Is a DOM optional, or MUST all conforming user agents implement the DOM?
- What version of DOM?
- Do we have to test the full DOM implementation of the host language in our conformance suite (hope not)?
- Can we write a conformance clause such that we require the DOM functionality on our information for conformance here? Or is the above statement what we need?

7.3.4. Accessibility API mapping

User agents **SHOULD** expose role, state, and property information provided by the author to accessibility APIs available in their operating platform. Refer to *Mapping States and Properties to Accessibility APIs* ([*ARIA-PRACTICES*], Section 12.1) for guidance about how to expose this information. This requirement parallels *User Agent Accessibility Guidelines 1.0 Section 6.3: Programmatic Access to non-HTML/XML Content* ([*UAAG*], Section 6.3), except that it applies even to HTML and XML content.

> **Note:** Not all platforms provide accessibility APIs, or provide interfaces that map to all the roles, states, and properties defined in this specification. User agents should expose those roles, states, and properties that are supported in order to support assistive technologies that work through the accessibility API. The remaining roles, states, and properties are available to assistive technologies via the DOM as per point 1 above, for those that provide explicit support for this specification.

Editorial Note: The working group is tracking issues around the question of how much about the API bindings should be normative and MUST, and how much they should be incorporated in this specification. These issues have not been resolved by the Working Group.

7.4. Assistive Technologies

This section is *informative*.

Assistive technologies **SHOULD** use available role, state and property information to present content to, and support interaction with, users in a manner, a user experience, appropriate to their users. This requirement parallels User Agent Accessibility Guidelines 1.0 Section 6.5: Programmatic operation of user agent user interface and *Section 6.6: Programmatic notification of changes* ([*UAAG*], Section 6.5 and 6.6) except that it applies to content, not just the user agent itself.

8. Appendices

This section is *informative*.

8.1. Implementations

8.1.1. Roles Implementation

Editorial note: the following references are used by the RDF but are not otherwise referenced in the document, and therefore are uncited. Explanations of these resources will be integrated into the prose.

[SKOS]
: *SKOS* is an area of work developing specifications and standards to support the use of knowledge organisation systems (KOS) such as thesauri, classification schemes, subject heading lists, taxonomies, terminologies, glossaries and other types of controlled vocabulary within the framework of the Semantic Web.

[DAISY]
: *DAISY* denotes the Digital Accessible Information System

[NIMAS]
: *NIMAS* the National Instructional Materials Accessibility Standard (NIMAS), is a voluntary standard to guide the production and electronic distribution of flexible digital instructional materials, such as textbooks, so that they can be more easily converted to Braille, text-to-speech, and other accessible formats.

[DTB]
: *The Digital Talking Book* standard that defines the format and content of the electronic file set that comprises a digital talking book (DTB) and establishes a limited set of requirements for DTB playback devices.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE rdf:RDF [
  <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">
  <!ENTITY dc "http://dublincore.org/2003/03/24/dces#">
  <!ENTITY owl "http://www.w3.org/2002/07/owl#">
  <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#">
  <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#">
  <!ENTITY states "http://www.w3.org/2005/07/aaa#">
]>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
         xmlns:role="http://www.w3.org/1999/xhtml/vocab#"
         xmlns:states="http://www.w3.org/2005/07/aaa#"
         xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
         xmlns:dc="http://purl.org/dc/elements/1.1/#"
         xmlns:owl="http://www.w3.org/2002/07/owl#"
         xml:base="http://www.w3.org/2005/01/wai-rdf/GUIRoleTaxonomy"><!--==Obje
      <rdfs:comment xml:lang="en">This is similar to type but without
                 inheritance of limitations and properties. role:baseConcepts a
                 a substitute for inheritance for external concepts. </rdfs:com
      <rdfs:subpropertyOf rdf:resource="rdfs:seeAlso"/>
   </owl:ObjectProperty>
   <owl:ObjectProperty rdf:ID="supportedState">
      <rdfs:comment xml:lang="en">A state that can be supported for this a
                 Role</rdfs:comment>
      <rdfs:domain rdf:resource="#roletype"/>
   </owl:ObjectProperty>
   <owl:ObjectProperty rdf:ID="scope">
      <rdfs:comment xml:lang="en">Context where this role is
                 allowed</rdfs:comment>
      <rdfs:domain rdf:resource="#roletype"/>
   </owl:ObjectProperty>
   <owl:ObjectProperty rdf:ID="mustContain">
      <rdfs:comment xml:lang="en">A child that must be contained by this
                 role</rdfs:comment>
      <rdfs:subpropertyOf rdf:resource="#scope"/>
   </owl:ObjectProperty>
   <owl:ObjectProperty rdf:ID="default">
      <rdfs:comment>Default value of a supported state in the context of
                 this role</rdfs:comment>
   </owl:ObjectProperty>
   <owl:ObjectProperty rdf:ID="nameFrom">
      <rdfs:comment>How a role type name is extracted and referenced
                 inside a document. Values are "author": name comes from values
                 provided by the author in explict markup features; and "subtre
                 name comes from the text value of the element node.</rdfs:comm
      <rdfs:domain rdf:resource="#widget"/>
   </owl:ObjectProperty>
   <owl:ObjectProperty rdf:ID="childrenArePresentational">
      <rdfs:comment xml:lang="en">The children are presenational. Assistive
                              technologies may choose to hid the children from
      <rdf:range rdf:resource="http://www.w3.org/2001/XMLSchema#boolean"/>
      <rdfs:domain rdf:resource="#roletype"/>
   </owl:ObjectProperty>
   <!--== Base Types ==--><owl:Class rdf:ID="roletype">
      <rdfs:subClassOf>
         <owl:Restriction>
            <owl:onProperty rdf:resource="http://dublincore.org/2003/03/24/dces#
            <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#stri
         </owl:Restriction>
      </rdfs:subClassOf>
      <rdfs:seeAlso rdf:resource="http://www.w3.org/TR/xhtml-role/#s_role_module
      <rdfs:seeAlso rdf:resource="http://www.w3.org/TR/html401/struct/links.html
      <rdfs:seeAlso rdf:resource="http://purl.org/dc/elements/1.1/type"/>
      <role:supportedState rdf:resource="http://www.w3.org/2005/07/aaa#atomic"/>
      <role:supportedState rdf:resource="http://www.w3.org/2005/07/aaa#busy"/>
      <role:supportedState rdf:resource="http://www.w3.org/2005/07/aaa#channel"/
      <role:supportedState rdf:resource="http://www.w3.org/2005/07/aaa#controls"
      <role:supportedState rdf:resource="http://www.w3.org/2005/07/aaa#described
      <role:supportedState rdf:resource="http://www.w3.org/2005/07/aaa#disabled"
      <role:supportedState rdf:resource="http://www.w3.org/2005/07/aaa#dropeffec
      <role:supportedState rdf:resource="http://www.w3.org/2005/07/aaa#flowto"/>
      <role:supportedState rdf:resource="http://www.w3.org/2005/07/aaa#grab"/>
      <role:supportedState rdf:resource="http://www.w3.org/2005/07/aaa#haspopup"
      <role:supportedState rdf:resource="http://www.w3.org/2005/07/aaa#hidden"/>
      <role:supportedState rdf:resource="http://www.w3.org/2005/07/aaa#invalid"/
      <role:supportedState rdf:resource="http://www.w3.org/2005/07/aaa#labelledb
      <role:supportedState rdf:resource="http://www.w3.org/2005/07/aaa#live"/>
      <role:supportedState rdf:resource="http://www.w3.org/2005/07/aaa#owns"/>
      <role:supportedState rdf:resource="http://www.w3.org/2005/07/aaa#relevant"
      <role:supportedState rdf:resource="http://www.w3.org/2005/07/aaa#required"
   </owl:Class>
   <owl:Class rdf:ID="widget">
      <rdfs:subClassOf rdf:resource="#roletype"/>
   </owl:Class>
   <owl:Class rdf:ID="structure">
      <rdfs:subClassOf rdf:resource="#roletype"/>
      <role:supportedState rdf:resource="http://www.w3.org/2005/07/aaa#expanded"
   </owl:Class>
   <owl:Class rdf:ID="composite">
      <rdfs:subClassOf rdf:resource="#widget"/>
      <role:supportedState rdf:resource="http://www.w3.org/2005/07/aaa#activedes
   </owl:Class>
   <owl:Class rdf:ID="window">
      <rdfs:subClassOf rdf:resource="#roletype"/>
      <role:supportedState rdf:resource="http://www.w3.org/2005/07/aaa#expanded"
   </owl:Class>
   <!--== User Input Widgets ==--><owl:Class rdf:ID="input">
      <rdfs:subClassOf rdf:resource="#widget"/>
      <rdfs:seeAlso rdf:resource="http://www.w3.org/TR/2007/REC-xforms-20071029/
   </owl:Class>
   <owl:Class rdf:ID="select">
      <rdfs:subClassOf rdf:resource="#composite"/>
      <rdfs:subClassOf rdf:resource="#group"/>
      <rdfs:subClassOf rdf:resource="#input"/>
   </owl:Class>
   <owl:Class rdf:ID="listbox">
      <rdfs:subClassOf rdf:resource="#list"/>
      <rdfs:subClassOf rdf:resource="#select"/>
      <rdfs:seeAlso rdf:resource="http://www.w3.org/TR/html401/interact/forms.ht
      <rdfs:seeAlso rdf:resource="http://www.w3.org/TR/2007/REC-xforms-20071029/
      <role:mustContain rdf:resource="#option"/>
      <role:supportedState rdf:resource="http://www.w3.org/2005/07/aaa#multisele
   </owl:Class>
   <owl:Class rdf:ID="combobox">
      <rdfs:subClassOf rdf:resource="#select"/>
      <rdfs:seeAlso rdf:resource="http://www.w3.org/TR/html401/interact/forms.ht
      <rdfs:seeAlso rdf:resource="http://www.w3.org/TR/2007/REC-xforms-20071029/
      <role:mustContain rdf:resource="#listbox"/>
   </owl:Class>
   <owl:Class rdf:ID="option">
      <rdfs:subClassOf rdf:resource="#input"/>
      <role:baseConcept rdf:resource="http://www.w3.org/TR/html4/interact/forms.
      <rdfs:seeAlso rdf:resource="#listitem"/>
      <rdfs:seeAlso rdf:resource="http://www.w3.org/TR/2007/REC-xforms-20071029/
      <role:supportedState rdf:resource="http://www.w3.org/2005/07/aaa#checked"/
      <role:supportedState rdf:resource="http://www.w3.org/2005/07/aaa#selected"
   </owl:Class>
   <owl:Class rdf:ID="checkbox">
      <rdfs:subClassOf rdf:resource="#input"/>
      <rdfs:seeAlso rdf:resource="http://www.w3.org/TR/html401/interact/forms.ht
      <rdfs:seeAlso rdf:resource="#option"/>
   </owl:Class>
   <owl:Class rdf:ID="radiogroup">
      <rdfs:subClassOf rdf:resource="#select"/>
      <role:mustContain rdf:resource="#radio"/>
   </owl:Class>
   <owl:Class rdf:ID="radio">
      <rdfs:subClassOf rdf:resource="#checkbox"/>
      <rdfs:seeAlso rdf:resource="http://www.w3.org/TR/html401/interact/forms.ht
   </owl:Class>
   <owl:Class rdf:ID="textbox">
      <rdfs:subClassOf rdf:resource="#input"/>
      <rdfs:seeAlso rdf:resource="http://www.w3.org/TR/2007/REC-xforms-20071029/
      <rdfs:seeAlso rdf:resource="http://www.w3.org/TR/html4/interact/forms.html
      <role:supportedState rdf:resource="http://www.w3.org/2005/07/aaa#autocompl
      <role:supportedState rdf:resource="http://www.w3.org/2005/07/aaa#multiline
      <role:supportedState rdf:resource="http://www.w3.org/2005/07/aaa#readonly"
   </owl:Class>
   <owl:Class rdf:ID="range">
      <rdfs:subClassOf rdf:resource="#input"/>
      <role:supportedState rdf:resource="http://www.w3.org/2005/07/aaa#valuemax"
      <role:supportedState rdf:resource="http://www.w3.org/2005/07/aaa#valuemin"
      <role:supportedState rdf:resource="http://www.w3.org/2005/07/aaa#valuenow"
      <role:supportedState rdf:resource="http://www.w3.org/2005/07/aaa#valuetext
   </owl:Class>
   <owl:Class rdf:ID="slider">
      <rdfs:subClassOf rdf:resource="#range"/>
   </owl:Class>
   <owl:Class rdf:ID="spinbutton">
      <rdfs:subClassOf rdf:resource="#composite"/>
      <rdfs:subClassOf rdf:resource="#range"/>
   </owl:Class>
   <!--== User Interface Elements ==--><owl:Class rdf:ID="button">
      <rdfs:subClassOf rdf:resource="#widget"/>
      <role:baseConcept rdf:resource="http://www.w3.org/TR/html4/interact/forms.
      <rdfs:seeAlso rdf:resource="#link"/>
      <rdfs:seeAlso rdf:resource="http://www.w3.org/TR/2007/REC-xforms-20071029/
      <role:supportedState rdf:resource="http://www.w3.org/2005/07/aaa#pressed"/
   </owl:Class>
   <owl:Class rdf:ID="link">
      <rdfs:subClassOf rdf:resource="#widget"/>
      <rdfs:seeAlso rdf:resource="http://www.w3.org/TR/html401/struct/links.html
   </owl:Class>
   <owl:Class rdf:ID="menu">
      <rdfs:subClassOf rdf:resource="#list"/>
      <rdfs:subClassOf rdf:resource="#select"/>
      <rdfs:seeAlso rdf:resource="http://www.loc.gov/nls/z3986/v100/dtbook110doc
      <rdfs:seeAlso rdf:resource="http://www.w3.org/TR/2007/REC-xforms-20071029/
      <rdfs:seeAlso rdf:resource="http://java.sun.com/j2se/1.3/docs/api/javax/ac
      <role:mustContain rdf:resource="#menuitem"/>
      <role:mustContain rdf:resource="#menuitemcheckbox"/>
      <role:mustContain rdf:resource="#menuitemradio"/>
      <role:mustContain rdf:resource="#separator"/>
   </owl:Class>
   <owl:Class rdf:ID="menubar">
      <rdfs:subClassOf rdf:resource="#group"/>
      <rdfs:seeAlso rdf:resource="#toolbar"/>
   </owl:Class>
   <owl:Class rdf:ID="toolbar">
      <rdfs:subClassOf rdf:resource="#group"/>
      <rdfs:seeAlso rdf:resource="#menubar"/>
   </owl:Class>
   <owl:Class rdf:ID="menuitem">
      <rdfs:subClassOf rdf:resource="#input"/>
      <rdfs:seeAlso rdf:resource="http://java.sun.com/j2se/1.3/docs/api/javax/ac
      <rdfs:seeAlso rdf:resource="#listitem"/>
      <rdfs:seeAlso rdf:resource="#option"/>
      <role:scope rdf:resource="#menu"/>
   </owl:Class>
   <owl:Class rdf:ID="menuitemcheckbox">
      <rdfs:subClassOf rdf:resource="#checkbox"/>
      <rdfs:subClassOf rdf:resource="#menuitem"/>
      <rdfs:seeAlso rdf:resource="#menuitem"/>
      <role:scope rdf:resource="#menu"/>
   </owl:Class>
   <owl:Class rdf:ID="menuitemradio">
      <rdfs:subClassOf rdf:resource="#menuitem"/>
      <rdfs:subClassOf rdf:resource="#radio"/>
      <rdfs:seeAlso rdf:resource="#menuitem"/>
      <role:scope rdf:resource="#menu"/>
   </owl:Class>
   <owl:Class rdf:ID="tooltip">
      <rdfs:subClassOf rdf:resource="#description"/>
   </owl:Class>
   <owl:Class rdf:ID="tabpanel">
      <rdfs:subClassOf rdf:resource="#region"/>
   </owl:Class>
   <owl:Class rdf:ID="tablist">
      <rdfs:subClassOf rdf:resource="#composite"/>
      <rdfs:subClassOf rdf:resource="#directory"/>
      <rdfs:seeAlso rdf:resource="http://www.daisy.org/z3986/2005/Z3986-2005.htm
      <role:mustContain rdf:resource="#tab"/>
   </owl:Class>
   <owl:Class rdf:ID="tab">
      <rdfs:subClassOf rdf:resource="#sectionhead"/>
      <rdfs:subClassOf rdf:resource="#widget"/>
      <role:scope rdf:resource="#tablist"/>
   </owl:Class>
   <owl:Class rdf:ID="tree">
      <rdfs:subClassOf rdf:resource="#select"/>
      <role:mustContain rdf:resource="#treeitem"/>
      <role:supportedState rdf:resource="http://www.w3.org/2005/07/aaa#multisele
   </owl:Class>
   <owl:Class rdf:ID="treeitem">
      <rdfs:subClassOf rdf:resource="#listitem"/>
      <rdfs:subClassOf rdf:resource="#option"/>
      <role:scope rdf:resource="#tree"/>
   </owl:Class>
   <!--== Document Structure ==--><owl:Class rdf:ID="section">
      <rdfs:subClassOf rdf:resource="#structure"/>
      <rdfs:seeAlso rdf:resource="http://www.loc.gov/nls/z3986/v100/dtbook110doc
      <rdfs:seeAlso rdf:resource="http://www.loc.gov/nls/z3986/v100/dtbook110doc
      <rdfs:seeAlso rdf:resource="http://www.w3.org/TR/REC-smil/#par"/>
   </owl:Class>
   <owl:Class rdf:ID="sectionhead">
      <rdfs:subClassOf rdf:resource="#structure"/>
   </owl:Class>
   <owl:Class rdf:ID="document">
      <rdfs:subClassOf rdf:resource="#structure"/>
      <rdfs:seeAlso rdf:resource="http://www.w3.org/TR/di-gloss/#def-delivery-un
   </owl:Class>
   <owl:Class rdf:ID="region">
      <rdfs:subClassOf rdf:resource="#section"/>
      <rdfs:seeAlso rdf:resource="http://www.w3.org/TR/html401/present/frames.ht
      <rdfs:seeAlso rdf:resource="http://www.w3.org/TR/di-gloss/#def-perceivable
      <rdfs:seeAlso rdf:resource="#section"/>
   </owl:Class>
   <owl:Class rdf:ID="heading">
      <rdfs:subClassOf rdf:resource="#sectionhead"/>
```

```
      <rdfs:seeAlso rdf:resource="http://www.w3.org/TR/html4/struct/global.html#
      <rdfs:seeAlso rdf:resource="http://www.w3.org/TR/html4/struct/global.html#
      <rdfs:seeAlso rdf:resource="http://www.w3.org/TR/html4/struct/global.html#
      <rdfs:seeAlso rdf:resource="http://www.w3.org/TR/html4/struct/global.html#
      <rdfs:seeAlso rdf:resource="http://www.w3.org/TR/html4/struct/global.html#
      <rdfs:seeAlso rdf:resource="http://www.w3.org/TR/html4/struct/global.html#
      <rdfs:seeAlso rdf:resource="http://www.loc.gov/nls/z3986/v100/dtbook110doc
      <role:supportedState rdf:resource="http://www.w3.org/2005/07/aaa#level"/>
   </owl:Class>
   <owl:Class rdf:ID="list">
      <rdfs:subClassOf rdf:resource="#region"/>
      <role:baseConcept rdf:resource="http://www.w3.org/TR/html4/struct/lists.ht
      <role:mustContain rdf:resource="#group"/>
      <role:mustContain rdf:resource="#listitem"/>
   </owl:Class>
   <owl:Class rdf:ID="listitem">
      <rdfs:subClassOf rdf:resource="#section"/>
      <role:baseConcept rdf:resource="http://www.w3.org/TR/html4/struct/lists.ht
      <rdfs:seeAlso rdf:resource="http://www.w3.org/TR/2007/REC-xforms-20071029/
      <role:scope rdf:resource="#list"/>
      <role:supportedState rdf:resource="http://www.w3.org/2005/07/aaa#level"/>
      <role:supportedState rdf:resource="http://www.w3.org/2005/07/aaa#posinset"
      <role:supportedState rdf:resource="http://www.w3.org/2005/07/aaa#setsize"/
   </owl:Class>
   <owl:Class rdf:ID="group">
      <rdfs:subClassOf rdf:resource="#section"/>
      <rdfs:seeAlso rdf:resource="http://www.w3.org/TR/html401/interact/forms.ht
      <role:supportedState rdf:resource="http://www.w3.org/2005/07/aaa#activedes
   </owl:Class>
   <owl:Class rdf:ID="grid">
      <rdfs:subClassOf rdf:resource="#composite"/>
      <rdfs:subClassOf rdf:resource="#section"/>
      <role:baseConcept rdf:resource="http://www.w3.org/TR/html4/struct/tables.h
      <role:mustContain rdf:resource="#row"/>
      <role:supportedState rdf:resource="http://www.w3.org/2005/07/aaa#level"/>
      <role:supportedState rdf:resource="http://www.w3.org/2005/07/aaa#multisele
      <role:supportedState rdf:resource="http://www.w3.org/2005/07/aaa#readonly"
   </owl:Class>
   <owl:Class rdf:ID="row">
      <rdfs:subClassOf rdf:resource="#group"/>
      <role:baseConcept rdf:resource="http://www.w3.org/TR/html4/struct/tables.h
      <role:scope rdf:resource="#grid"/>
      <role:scope rdf:resource="#treegrid"/>
      <role:mustContain rdf:resource="#gridcell"/>
      <role:supportedState rdf:resource="http://www.w3.org/2005/07/aaa#level"/>
      <role:supportedState rdf:resource="http://www.w3.org/2005/07/aaa#selected"
   </owl:Class>
   <owl:Class rdf:ID="gridcell">
      <rdfs:subClassOf rdf:resource="#section"/>
      <rdfs:subClassOf rdf:resource="#widget"/>
      <role:baseConcept rdf:resource="http://www.w3.org/TR/html4/struct/tables.h
      <role:scope rdf:resource="#row"/>
      <role:supportedState rdf:resource="http://www.w3.org/2005/07/aaa#level"/>
      <role:supportedState rdf:resource="http://www.w3.org/2005/07/aaa#readonly"
      <role:supportedState rdf:resource="http://www.w3.org/2005/07/aaa#selected"
   </owl:Class>
   <owl:Class rdf:ID="rowheader">
      <rdfs:subClassOf rdf:resource="#gridcell"/>
      <rdfs:subClassOf rdf:resource="#sectionhead"/>
      <role:baseConcept rdf:resource="http://www.w3.org/TR/html4/struct/tables.h
      <role:scope rdf:resource="#row"/>
      <role:supportedState rdf:resource="http://www.w3.org/2005/07/aaa#sort"/>
   </owl:Class>
   <owl:Class rdf:ID="columnheader">
      <rdfs:subClassOf rdf:resource="#gridcell"/>
   <rdfs:subClassOf rdf:resource="#sectionhead"/>
   <role:baseConcept rdf:resource="http://www.w3.org/TR/html4/struct/tables.h
   <role:scope rdf:resource="#row"/>
   <role:supportedState rdf:resource="http://www.w3.org/2005/07/aaa#sort"/>
</owl:Class>
<owl:Class rdf:ID="treegrid">
   <rdfs:subClassOf rdf:resource="#grid"/>
   <rdfs:subClassOf rdf:resource="#tree"/>
   <role:mustContain rdf:resource="#row"/>
</owl:Class>
<owl:Class rdf:ID="description">
   <rdfs:subClassOf rdf:resource="#section"/>
</owl:Class>
<owl:Class rdf:ID="directory">
   <rdfs:subClassOf rdf:resource="#list"/>
   <rdfs:seeAlso rdf:resource="http://www.daisy.org/z3986/2005/Z3986-2005.htm
</owl:Class>
<owl:Class rdf:ID="img">
   <rdfs:subClassOf rdf:resource="#section"/>
   <rdfs:seeAlso rdf:resource="http://www.loc.gov/nls/z3986/v100/dtbook110doc
</owl:Class>
<owl:Class rdf:ID="presentation">
   <rdfs:subClassOf rdf:resource="#structure"/>
</owl:Class>
<owl:Class rdf:ID="separator">
   <rdfs:subClassOf rdf:resource="#structure"/>
   <rdfs:seeAlso rdf:resource="http://www.w3.org/TR/html401/present/graphics.
</owl:Class>
<owl:Class rdf:ID="math">
   <rdfs:subClassOf rdf:resource="#section"/>
</owl:Class>
<!--== Specialized Regions ==--><owl:Class rdf:ID="application">
   <rdfs:subClassOf rdf:resource="#region"/>
   <rdfs:seeAlso rdf:resource="http://www.w3.org/TR/di-gloss/#def-delivery-un
</owl:Class>
<owl:Class rdf:ID="dialog">
   <rdfs:subClassOf rdf:resource="#window"/>
</owl:Class>
<owl:Class rdf:ID="alert">
   <rdfs:subClassOf rdf:resource="#region"/>
   <rdfs:seeAlso rdf:resource="http://www.w3.org/TR/2007/REC-xforms-20071029/
</owl:Class>
<owl:Class rdf:ID="alertdialog">
   <rdfs:subClassOf rdf:resource="#alert"/>
   <rdfs:subClassOf rdf:resource="#dialog"/>
   <rdfs:seeAlso rdf:resource="http://www.w3.org/TR/2007/REC-xforms-20071029/
</owl:Class>
<owl:Class rdf:ID="marquee">
   <rdfs:subClassOf rdf:resource="#section"/>
</owl:Class>
<owl:Class rdf:ID="log">
   <rdfs:subClassOf rdf:resource="#region"/>
</owl:Class>
<owl:Class rdf:ID="status">
   <rdfs:subClassOf rdf:resource="#composite"/>
   <rdfs:subClassOf rdf:resource="#region"/>
</owl:Class>
<owl:Class rdf:ID="progressbar">
   <rdfs:subClassOf rdf:resource="#widget"/>
   <rdfs:seeAlso rdf:resource="#status"/>
   <role:supportedState rdf:resource="http://www.w3.org/2005/07/aaa#valuemax"
   <role:supportedState rdf:resource="http://www.w3.org/2005/07/aaa#valuemin"
   <role:supportedState rdf:resource="http://www.w3.org/2005/07/aaa#valuenow"
   <role:supportedState rdf:resource="http://www.w3.org/2005/07/aaa#valuetext
</owl:Class>
```

```
   <owl:Class rdf:ID="timer">
      <rdfs:subClassOf rdf:resource="#status"/>
   </owl:Class>
   <!--== Landmark Roles Inherited from the XHTML Role Attribute Module ==--><ow
      <rdfs:subClassOf rdf:resource="#document"/>
      <rdfs:subClassOf rdf:resource="#region"/>
      <rdfs:seeAlso rdf:resource="http://www.w3.org/TR/2008/WD-html5-20080122/#t
   </owl:Class>
   <owl:Class rdf:ID="banner">
      <rdfs:subClassOf rdf:resource="#region"/>
   </owl:Class>
   <owl:Class rdf:ID="complementary">
      <rdfs:subClassOf rdf:resource="#region"/>
   </owl:Class>
   <owl:Class rdf:ID="contentinfo">
      <rdfs:subClassOf rdf:resource="#region"/>
   </owl:Class>
   <owl:Class rdf:ID="main">
      <rdfs:subClassOf rdf:resource="#region"/>
   </owl:Class>
   <owl:Class rdf:ID="navigation">
      <rdfs:subClassOf rdf:resource="#region"/>
   </owl:Class>
   <owl:Class rdf:ID="search">
      <rdfs:subClassOf rdf:resource="#region"/>
   </owl:Class>
   <!--== Section Roles inherited from XHTML Role Attribute Module ==--><owl:Cla
      <rdfs:subClassOf rdf:resource="#section"/>
   </owl:Class>
   <owl:Class rdf:ID="note">
      <rdfs:subClassOf rdf:resource="#section"/>
   </owl:Class>
</rdf:RDF>
```

8.1.2. ARIA Attributes Module

This module declares the ARIA attributes as a module that can be included in a modularlized DTD. A sample XHTML DTD using this module follows. Note the ARIA attributes are in no namespace, and the attribute name begins with "aria-" to reduce the likelihood of collision with existing attributes.

```
<!-- ............................................................... -->
<!-- ARIA Attributes Module ........................................ -->
<!-- file: aria-attributes.mod

     This is ARIA Attributes - the Accessible Rick Internat Applications
     attributes module for XHTML.

     Copyright 2008 W3C (MIT, ERCIM, Keio), All Rights Reserved.

     This DTD module is identified by the PUBLIC and SYSTEM identifiers:

       PUBLIC "-//W3C//ENTITIES XHTML ARIA Attributes 1.0//EN"
       SYSTEM "http://www.w3.org/2005/07/aaa/aria-attributes.mod"

     Revisions:
     (none)
     ................................................................ -->

<!-- states -->
<!ENTITY % ARIA.states.attrib "
    aria-busy ( true | false | error ) 'false'
    aria-checked ( true | false | mixed ) 'undefined'
```

```
        aria-disabled ( true | false ) 'false'
        aria-dropeffect NMTOKENS 'none'
        aria-expanded ( true | false | undefined ) 'undefined'
        aria-grab ( true | supported | false ) 'false'
        aria-hidden ( true | false ) 'false'
        aria-invalid ( true | false ) 'false'
        aria-pressed ( true | false | mixed | undefined ) 'undefined'
        aria-selected ( true | false | undefined ) 'undefined'
    ">

    <!-- properties -->
    <!ENTITY % ARIA.props.attrib "
        aria-activedescendant IDREF #IMPLIED
        aria-atomic ( true | false ) 'false'
        aria-autocomplete ( inline | list | both | none ) 'none'
        aria-channel ( main | notify ) 'main'
        aria-controls IDREFS #IMPLIED
        aria-describedby IDREFS #IMPLIED
        aria-flowto IDREFS #IMPLIED
        aria-haspopup ( true | false ) 'false'
        aria-labelledby IDREFS #IMPLIED
        aria-level CDATA #IMPLIED
        aria-live ( off | polite | assertive | rude ) 'off'
        aria-multiline ( true | false ) 'false'
        aria-multiselectable ( true | false ) 'false'
        aria-owns IDREFS #IMPLIED
        aria-posinset CDATA #IMPLIED
        aria-readonly ( true | false ) 'false'
        aria-relevant NMTOKENS 'additions text'
        aria-required ( true | false ) 'false'
        aria-setsize CDATA #IMPLIED
        aria-sort ( ascending | descending | none | other ) 'none'
        aria-valuemax CDATA #IMPLIED
        aria-valuemin CDATA #IMPLIED
        aria-valuenow CDATA #IMPLIED
        aria-valuetext CDATA #IMPLIED
    ">

    <!ENTITY % ARIA.extra.attrib "" >

    <!ENTITY % ARIA.attrib "
        %ARIA.states.attrib;
        %ARIA.props.attrib;
        %ARIA.extra.attrib;
    ">

    <!-- End aria-attributes Module ..................................
```

8.1.3. Sample XHTML plus ARIA DTD

This sample DTD extends XHTML 1.1 adds the ARIA state and property attributes to all its elements. It also adds the `tabindex` attribute to a wider set of elements, as a way of providing keyboard focus support.

```
<!-- ....................................................................... -->
<!-- XHTML+ARIA DTD  ....................................................... -->
<!-- file: xhtml-aria-1.dtd
-->

<!-- XHTML 1.1 + ARIA DTD

     This is an example markup language combining XHTML 1.1 and the ARIA
     modules.
     XHTML+ARIA
     Copyright 1998-2008 World Wide Web Consortium
        (Massachusetts Institute of Technology, European Research Consortium
         for Informatics and Mathematics, Keio University).
         All Rights Reserved.

     Permission to use, copy, modify and distribute the XHTML DTD and its
     accompanying documentation for any purpose and without fee is hereby
     granted in perpetuity, provided that the above copyright notice and
     this paragraph appear in all copies.  The copyright holders make no
     representation about the suitability of the DTD for any purpose.

     It is provided "as is" without expressed or implied warranty.

-->
<!-- This is the driver file for a sample XHTML + ARIA DTD.

     Please use this public identifier to identify it:

         "-//W3C//DTD XHTML+ARIA EXAMPLE 1.0//EN"
-->
<!ENTITY % XHTML.version  "XHTML+ARIA 1.0" >

<!-- Use this URI to identify the default namespace:

         "http://www.w3.org/1999/xhtml"

     See the Qualified Names module for information
     on the use of namespace prefixes in the DTD.

         Note that XHTML namespace elements are not prefixed by default,
         but the XHTML namespace prefix is defined as "xhtml" so that
         other markup languages can extend this one and use the XHTML
         prefixed global attributes if required.

-->
<!ENTITY % NS.prefixed "IGNORE" >
<!ENTITY % XHTML.prefix "xhtml" >

<!-- Be sure to include prefixed global attributes - we don't need
     them, but languages that extend XHTML 1.1 might.
-->
<!ENTITY % XHTML.global.attrs.prefixed "INCLUDE" >

<!-- Reserved for use with the XLink namespace:
-->
<!ENTITY % XLINK.xmlns "" >
<!ENTITY % XLINK.xmlns.attrib "" >

<!-- For example, if you are using XHTML 1.1 directly, use the public
     identifier in the DOCTYPE declaration, with the namespace declaration
     on the document element to identify the default namespace:

       <?xml version="1.0"?>
       <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML+ARIA 1.0//EN"
                             "http://www.w3.org/MarkUp/DTD/xhtml-aria-1.dtd">
       <html xmlns="http://www.w3.org/1999/xhtml"
             xml:lang="en">
       ...
       </html>

     Revisions:
     (none)
-->
```

```
<!-- reserved for future use with document profiles -->
<!ENTITY % XHTML.profile  "" >

<!-- ensure XHTML Notations are disabled -->
<!ENTITY % xhtml-notations.module "IGNORE" >

<!-- Bidirectional Text features
     This feature-test entity is used to declare elements
     and attributes used for bidirectional text support.
-->
<!ENTITY % XHTML.bidi  "INCLUDE" >

<!-- :::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::: -->

<!-- Pre-Framework Redeclaration placeholder  .................... -->
<!-- this serves as a location to insert markup declarations
     into the DTD prior to the framework declarations.
-->
<!ENTITY % xhtml-prefw-redecl.module "IGNORE" >
<!ENTITY % xhtml-prefw-redecl.mod "" >
<![%xhtml-prefw-redecl.module;[
%xhtml-prefw-redecl.mod;
<!-- end of xhtml-prefw-redecl.module -->]]>

<!-- we need the datatypes now -->
<!ENTITY % xhtml-datatypes.module "INCLUDE" >
<![%xhtml-datatypes.module;[
<!ENTITY % xhtml-datatypes.mod
     PUBLIC "-//W3C//ENTITIES XHTML Datatypes 1.0//EN"
            "http://www.w3.org/MarkUp/DTD/xhtml-datatypes-1.mod" >
%xhtml-datatypes.mod;]]>

<!-- bring in the ARIA attributes cause we need them in Common -->
<!ENTITY % ARIA-attributes.module "INCLUDE" >
<![%ARIA-attributes.module;[
<!ENTITY % ARIA-attributes.mod
     PUBLIC "-//W3C//ENTITIES XHTML ARIA Attributes 1.0//EN"
            "http://www.w3.org/2005/07/aaa/aria-attributes-1.mod" >
%ARIA-attributes.mod;]]>

<!-- bring in the role attribute cause it goes in Common too -->
<!ENTITY % xhtml-role.module "INCLUDE" >
<![%xhtml-role.module;[
<!ENTITY % xhtml-role.mod
     PUBLIC "-//W3C//ENTITIES XHTML Role Attribute 1.0//EN"
            "http://www.w3.org/MarkUp/DTD/xhtml-role-qname-1.mod" >
%xhtml-role.mod;]]>

<!ENTITY % xhtml-events.module "INCLUDE" >

<!-- populate common.extra -->
<!ENTITY % Common.extra.attrib
   "tabindex     %Number.datatype;        #IMPLIED
    %xhtml-role.attrs.qname;
    %ARIA.attrib;"
>

<!-- Inline Style Module  ........................................ -->
<!ENTITY % xhtml-inlstyle.module "INCLUDE" >
<![%xhtml-inlstyle.module;[
<!ENTITY % xhtml-inlstyle.mod
     PUBLIC "-//W3C//ELEMENTS XHTML Inline Style 1.0//EN"
            "http://www.w3.org/MarkUp/DTD/xhtml-inlstyle-1.mod" >
%xhtml-inlstyle.mod;]]>
```

```
<!-- declare Document Model module instantiated in framework
-->
<!ENTITY % xhtml-model.mod
     PUBLIC "-//W3C//ENTITIES XHTML 1.1 Document Model 1.0//EN"
            "http://www.w3.org/MarkUp/DTD/xhtml11-model-1.mod" >

<!-- Modular Framework Module (required) ........................ -->
<!ENTITY % xhtml-framework.module "INCLUDE" >
<![%xhtml-framework.module;[
<!ENTITY % xhtml-framework.mod
     PUBLIC "-//W3C//ENTITIES XHTML Modular Framework 1.0//EN"
            "http://www.w3.org/MarkUp/DTD/xhtml-framework-1.mod" >
%xhtml-framework.mod;]]>

<!-- Post-Framework Redeclaration placeholder  .................. -->
<!-- this serves as a location to insert markup declarations
     into the DTD following the framework declarations.
-->
<!ENTITY % xhtml-postfw-redecl.module "IGNORE" >
<!ENTITY % xhtml-postfw-redecl.mod "">
<![%xhtml-postfw-redecl.module;[
%xhtml-postfw-redecl.mod;
<!-- end of xhtml-postfw-redecl.module -->]]>



<!-- Text Module (Required)  ..................................... -->
<!ENTITY % xhtml-text.module "INCLUDE" >
<![%xhtml-text.module;[
<!ENTITY % xhtml-text.mod
     PUBLIC "-//W3C//ELEMENTS XHTML Text 1.0//EN"
            "http://www.w3.org/MarkUp/DTD/xhtml-text-1.mod" >
%xhtml-text.mod;]]>

<!-- Hypertext Module (required) ................................. -->
<!ENTITY % a.attlist  "IGNORE" >
<!ENTITY % xhtml-hypertext.module "INCLUDE" >
<![%xhtml-hypertext.module;[
<!ENTITY % xhtml-hypertext.mod
     PUBLIC "-//W3C//ELEMENTS XHTML Hypertext 1.0//EN"
            "http://www.w3.org/MarkUp/DTD/xhtml-hypertext-1.mod" >
%xhtml-hypertext.mod;]]>
<!ATTLIST %a.qname;
      %Common.attrib;
      href         %URI.datatype;           #IMPLIED
      charset      %Charset.datatype;       #IMPLIED
      type         %ContentType.datatype;   #IMPLIED
      hreflang     %LanguageCode.datatype;  #IMPLIED
      rel          %LinkTypes.datatype;     #IMPLIED
      rev          %LinkTypes.datatype;     #IMPLIED
      accesskey    %Character.datatype;     #IMPLIED
>

<!-- Lists Module (required)  .................................... -->
<!ENTITY % xhtml-list.module "INCLUDE" >
<![%xhtml-list.module;[
<!ENTITY % xhtml-list.mod
     PUBLIC "-//W3C//ELEMENTS XHTML Lists 1.0//EN"
            "http://www.w3.org/MarkUp/DTD/xhtml-list-1.mod" >
%xhtml-list.mod;]]>

<!-- :::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::: -->

<!-- Edit Module  ................................................ -->
```

```
<!ENTITY % xhtml-edit.module "INCLUDE" >
<![%xhtml-edit.module;[
<!ENTITY % xhtml-edit.mod
     PUBLIC "-//W3C//ELEMENTS XHTML Editing Elements 1.0//EN"
            "http://www.w3.org/MarkUp/DTD/xhtml-edit-1.mod" >
%xhtml-edit.mod;]]>

<!-- BIDI Override Module  ...................................... -->
<!ENTITY % xhtml-bdo.module "%XHTML.bidi;" >
<![%xhtml-bdo.module;[
<!ENTITY % xhtml-bdo.mod
     PUBLIC "-//W3C//ELEMENTS XHTML BIDI Override Element 1.0//EN"
            "http://www.w3.org/MarkUp/DTD/xhtml-bdo-1.mod" >
%xhtml-bdo.mod;]]>

<!-- Ruby Module  ................................................ -->
<!ENTITY % Ruby.common.attlists "INCLUDE" >
<!ENTITY % Ruby.common.attrib "%Common.attrib;" >
<!ENTITY % xhtml-ruby.module "INCLUDE" >
<![%xhtml-ruby.module;[
<!ENTITY % xhtml-ruby.mod
     PUBLIC "-//W3C//ELEMENTS XHTML Ruby 1.0//EN"
            "http://www.w3.org/TR/ruby/xhtml-ruby-1.mod" >
%xhtml-ruby.mod;]]>

<!-- Presentation Module  ........................................ -->
<!ENTITY % xhtml-pres.module "INCLUDE" >
<![%xhtml-pres.module;[
<!ENTITY % xhtml-pres.mod
     PUBLIC "-//W3C//ELEMENTS XHTML Presentation 1.0//EN"
            "http://www.w3.org/MarkUp/DTD/xhtml-pres-1.mod" >
%xhtml-pres.mod;]]>

<!-- Link Element Module  ........................................ -->
<!ENTITY % xhtml-link.module "INCLUDE" >
<![%xhtml-link.module;[
<!ENTITY % xhtml-link.mod
     PUBLIC "-//W3C//ELEMENTS XHTML Link Element 1.0//EN"
            "http://www.w3.org/MarkUp/DTD/xhtml-link-1.mod" >
%xhtml-link.mod;]]>

<!-- Document Metainformation Module  ............................ -->
<!ENTITY % xhtml-meta.module "INCLUDE" >
<![%xhtml-meta.module;[
<!ENTITY % xhtml-meta.mod
     PUBLIC "-//W3C//ELEMENTS XHTML Metainformation 1.0//EN"
            "http://www.w3.org/MarkUp/DTD/xhtml-meta-1.mod" >
%xhtml-meta.mod;]]>

<!-- Base Element Module  ........................................ -->
<!ENTITY % xhtml-base.module "INCLUDE" >
<![%xhtml-base.module;[
<!ENTITY % xhtml-base.mod
     PUBLIC "-//W3C//ELEMENTS XHTML Base Element 1.0//EN"
            "http://www.w3.org/MarkUp/DTD/xhtml-base-1.mod" >
%xhtml-base.mod;]]>

<!-- Scripting Module  ........................................... -->
<!ENTITY % xhtml-script.module "INCLUDE" >
<![%xhtml-script.module;[
<!ENTITY % xhtml-script.mod
     PUBLIC "-//W3C//ELEMENTS XHTML Scripting 1.0//EN"
            "http://www.w3.org/MarkUp/DTD/xhtml-script-1.mod" >
%xhtml-script.mod;]]>
```

```
<!-- Style Sheets Module  ......................................... -->
<!ENTITY % xhtml-style.module "INCLUDE" >
<![%xhtml-style.module;[
<!ENTITY % xhtml-style.mod
     PUBLIC "-//W3C//ELEMENTS XHTML Style Sheets 1.0//EN"
            "http://www.w3.org/MarkUp/DTD/xhtml-style-1.mod" >
%xhtml-style.mod;]]>

<!-- Image Module  ............................................... -->
<!ENTITY % xhtml-image.module "INCLUDE" >
<![%xhtml-image.module;[
<!ENTITY % xhtml-image.mod
     PUBLIC "-//W3C//ELEMENTS XHTML Images 1.0//EN"
            "http://www.w3.org/MarkUp/DTD/xhtml-image-1.mod" >
%xhtml-image.mod;]]>

<!-- Client-side Image Map Module  ............................... -->
<!ENTITY % area.attlist "IGNORE" >
<!ENTITY % xhtml-csismap.module "INCLUDE" >
<![%xhtml-csismap.module;[
<!ENTITY % xhtml-csismap.mod
     PUBLIC "-//W3C//ELEMENTS XHTML Client-side Image Maps 1.0//EN"
            "http://www.w3.org/MarkUp/DTD/xhtml-csismap-1.mod" >
%xhtml-csismap.mod;]]>
<!ATTLIST %area.qname;
      %Common.attrib;
      href         %URI.datatype;            #IMPLIED
      shape        %Shape.datatype;          'rect'
      coords       %Coords.datatype;         #IMPLIED
      nohref       ( nohref )                #IMPLIED
      alt          %Text.datatype;           #REQUIRED
      accesskey    %Character.datatype;      #IMPLIED
>

<!-- Server-side Image Map Module  ............................... -->
<!ENTITY % xhtml-ssismap.module "INCLUDE" >
<![%xhtml-ssismap.module;[
<!ENTITY % xhtml-ssismap.mod
     PUBLIC "-//W3C//ELEMENTS XHTML Server-side Image Maps 1.0//EN"
            "http://www.w3.org/MarkUp/DTD/xhtml-ssismap-1.mod" >
%xhtml-ssismap.mod;]]>

<!-- Param Element Module  ....................................... -->
<!ENTITY % xhtml-param.module "INCLUDE" >
<![%xhtml-param.module;[
<!ENTITY % xhtml-param.mod
     PUBLIC "-//W3C//ELEMENTS XHTML Param Element 1.0//EN"
            "http://www.w3.org/MarkUp/DTD/xhtml-param-1.mod" >
%xhtml-param.mod;]]>

<!-- Embedded Object Module  ..................................... -->
<!-- override the attributes -->
<!ENTITY % object.attlist "IGNORE" >
<!ENTITY % xhtml-object.module "INCLUDE" >
<![%xhtml-object.module;[
<!ENTITY % xhtml-object.mod
     PUBLIC "-//W3C//ELEMENTS XHTML Embedded Object 1.0//EN"
            "http://www.w3.org/MarkUp/DTD/xhtml-object-1.mod" >
%xhtml-object.mod;]]>

<!ATTLIST %object.qname;
      %Common.attrib;
      declare      ( declare )               #IMPLIED
      classid      %URI.datatype;            #IMPLIED
      codebase     %URI.datatype;            #IMPLIED
      data         %URI.datatype;            #IMPLIED
      type         %ContentType.datatype;    #IMPLIED
      codetype     %ContentType.datatype;    #IMPLIED
      archive      %URIs.datatype;           #IMPLIED
      standby      %Text.datatype;           #IMPLIED
      height       %Length.datatype;         #IMPLIED
      width        %Length.datatype;         #IMPLIED
      name         CDATA                     #IMPLIED
>

<!-- Tables Module ............................................... -->
<!ENTITY % xhtml-table.module "INCLUDE" >
<![%xhtml-table.module;[
<!ENTITY % xhtml-table.mod
     PUBLIC "-//W3C//ELEMENTS XHTML Tables 1.0//EN"
            "http://www.w3.org/MarkUp/DTD/xhtml-table-1.mod" >
%xhtml-table.mod;]]>

<!-- Forms Module  .............................................. -->

<!-- override attribute definitions to accomodate tabindex in Common -->


<!ENTITY % input.attlist "IGNORE" >
<!ENTITY % select.attlist "IGNORE" >
<!ENTITY % textarea.attlist "IGNORE" >
<!ENTITY % button.attlist "IGNORE" >
<!ENTITY % xhtml-form.module "INCLUDE" >
<![%xhtml-form.module;[
<!ENTITY % xhtml-form.mod
     PUBLIC "-//W3C//ELEMENTS XHTML Forms 1.0//EN"
            "http://www.w3.org/MarkUp/DTD/xhtml-form-1.mod" >
%xhtml-form.mod;]]>

<!ATTLIST %button.qname;
      %Common.attrib;
      name         CDATA                     #IMPLIED
      value        CDATA                     #IMPLIED
      type         ( button | submit | reset ) 'submit'
      disabled     ( disabled )              #IMPLIED
      accesskey    %Character.datatype;      #IMPLIED
>
<!ATTLIST %textarea.qname;
      %Common.attrib;
      name         CDATA                     #IMPLIED
      rows         %Number.datatype;         #REQUIRED
      cols         %Number.datatype;         #REQUIRED
      disabled     ( disabled )              #IMPLIED
      readonly     ( readonly )              #IMPLIED
      accesskey    %Character.datatype;      #IMPLIED
>
<!ATTLIST %select.qname;
      %Common.attrib;
      name         CDATA                     #IMPLIED
      size         %Number.datatype;         #IMPLIED
      multiple     ( multiple )              #IMPLIED
      disabled     ( disabled )              #IMPLIED
>
<!ENTITY % InputType.class
     "( text | password | checkbox | radio | submit
      | reset | file | hidden | image | button )"
>
<!ATTLIST %input.qname;
      %Common.attrib;
      type         %InputType.class;         'text'
      name         CDATA                    #IMPLIED
      value        CDATA                    #IMPLIED
      checked      ( checked )              #IMPLIED
      disabled     ( disabled )             #IMPLIED
      readonly     ( readonly )             #IMPLIED
      size         %Number.datatype;        #IMPLIED
      maxlength    %Number.datatype;        #IMPLIED
      src          %URI.datatype;           #IMPLIED
      alt          %Text.datatype;          #IMPLIED
      accesskey    %Character.datatype;     #IMPLIED
      accept       %ContentTypes.datatype;  #IMPLIED
>

<!-- Target Attribute Module  .................................... -->
<!ENTITY % xhtml-target.module "INCLUDE" >
<![%xhtml-target.module;[
<!ENTITY % xhtml-target.mod
     PUBLIC "-//W3C//ELEMENTS XHTML Target 1.0//EN"
            "http://www.w3.org/MarkUp/DTD/xhtml-target-1.mod" >
%xhtml-target.mod;]]>

<!-- Legacy Markup ............................................... -->
<!ENTITY % xhtml-legacy.module "IGNORE" >
<![%xhtml-legacy.module;[
<!ENTITY % xhtml-legacy.mod
     PUBLIC "-//W3C//ELEMENTS XHTML Legacy Markup 1.0//EN"
            "http://www.w3.org/MarkUp/DTD/xhtml-legacy-1.mod" >
%xhtml-legacy.mod;]]>

<!-- Document Structure Module (required)  ....................... -->
<!ENTITY % xhtml-struct.module "INCLUDE" >
<![%xhtml-struct.module;[
<!ENTITY % xhtml-struct.mod
     PUBLIC "-//W3C//ELEMENTS XHTML Document Structure 1.0//EN"
            "http://www.w3.org/MarkUp/DTD/xhtml-struct-1.mod" >
%xhtml-struct.mod;]]>
<!ENTITY % profile.attrib
     "profile      %URI.datatype;           '%XHTML.profile;'"
>
<!ENTITY % XHTML.version.attrib
     "version      %FPI.datatype;           #FIXED '%XHTML.version;'"
>

<!-- end of XHTML-ARIA DTD  ............................................. -->
<!-- ................................................................... -->
```

8.2. Glossary

While some terms are defined in place, the following definitions are used throughout this document. Familiarity with W3C XHTML 1.1 Recommendation [XHTML] and the W3C XML 1.0 Recommendation [XML] is highly recommended to understand these definitions.

Accessibility API

Operating systems and other platforms provide a set of interfaces that expose information about objects and events to assistive technology. Assistive technology uses these interfaces to get information about and interact with those widgets. Examples of this are the Java Accessibility API [JAPI], Microsoft Active Accessibility [MSAA], Apple Accessibility for COCOA [AAC], the Gnome Accessibility Toolkit (ATK) [ATK], and IAccessible2 [IA2].

Assistive Technology

Hardware and/or software that acts as a user agent, or along with a mainstream user agent, to provide services to meet the requirements of users with disabilities that go beyond those offered by the mainstream user agents.

Services provided by assistive technology include alternative presentations (e.g., as synthesized speech or magnified content), alternative input methods (e.g., speech recognition), additional navigation or orientation mechanisms, and content transformations (e.g., to make tables more accessible).

Assistive technologies often communicate data and messages with mainstream user agents by using and monitoring APIs.

The distinction between mainstream user agents and assistive technologies is not absolute. Many mainstream user agents provide some features to assist individuals with disabilities. The basic difference is that mainstream user agents target broad and diverse audiences that usually include people with and without disabilities. Assistive technologies target narrowly defined populations of users with specific disabilities. The assistance provided by an assistive technology is more specific and appropriate to the needs of its target users. The mainstream user agent may provide important services to assistive technologies like retrieving Web content from program objects or parsing markup into identifiable bundles.

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

- screen magnifiers, and other visual reading assistants, which are used by people with visual, perceptual and physical print disabilities to change text font, size, spacing, color, synchronization with speech, etc. in order to improve the visual readability of rendered text and images;
- screen readers, which are used by people who are blind to read textual information through synthesized speech or braille;
- text-to-speech software, which is used by some people with cognitive, language, and learning disabilities to convert text into synthetic speech;
- voice recognition software, which may be used by people who have some physical disabilities;
- alternative keyboards, which are used by people with certain physical disabilities to simulate the keyboard (including alternate keyboards that use headpointers, single switches, sip/puff and other special input devices.);
- alternative pointing devices, which are used by people with certain physical disabilities to simulate mouse pointing and button activations

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.

Characteristic

An assertion which can be a constraint.

Class

An abstract type of object.

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.

Managed State

@@

Namespace

A collection of related names. Qualifying a name in terms of the namespace to which it belongs allows these names to be distinguished from names in other namespaces that are spelled alike. Namespaces in this document are used in accordance with Namespaces in XML [XML-NAMES].

Object

A "thing" in the perceptual user experience, instantiated in markup languages by one or more elements, and converted into the object-oriented pattern by user agents. Objects are instances of abstract classes, which defines the general characteristics of object instances. A single DOM object may or may not correspond with a single object in accessibility API. An object in accessibility API encapsulates one or more DOM objects.

Property

Attributes that are essential to the nature of a given object. As such, they are less likely to change than states; a change of a property may significantly impact the meaning or presentation of an object. Properties mainly provide limitations on objects from the most general case implied by roles without properties applied.

Relationship

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

Role

An indicator of type. The object's role is the class of objects of which it is a member. 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.

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.

User Agent

Any software that retrieves and renders Web content for users, such as Web browsers, media players, plug-ins, and other programs including assistive technologies that help retrieve and render Web content.

Value

A literal that concretizes the information expressed by a state or property, or text content.

Widget

@@

8.3. References

[AAC]
Apple Accessibility for Cocoa. Available at: http://developer.apple.com /documentation/Cocoa/Conceptual/Accessibility/index.html.

[ARIA-PRACTICES]
WAI-ARIA Best Practices. L. Pappas, R. Schwerdtfeger, M. Cooper, Editors, W3C Working Draft (work in progress), 4 February 2008. This version of WAI-ARIA Best Practices is available at http://www.w3.org/TR/2008/WD-wai-aria-practices-20080204/. Latest version of WAI-ARIA Best Practices available at http://www.w3.org/TR/wai-aria-practices/.

[ARIA-PRIMER]
WAI-ARIA Primer. L. Pappas, R. Schwerdtfeger, M. Cooper, Editors, W3C Working Draft (work in progress), 4 February 2008. This version of WAI-ARIA Primer is available at http://www.w3.org/TR/2008/WD-wai-aria-primer-20080204/. Latest version of WAI-ARIA Primer available at http://www.w3.org/TR/wai-aria-primer/.

[ARIA-ROADMAP]
Roadmap for Accessible Rich Internet Applications (WAI-ARIA Roadmap), R. Schwerdtfeger, Editor, W3C Working Draft (work in progress), 4 February 2008. This version of WAI-ARIA Roadmap is available at http://www.w3.org/TR/2008/WD-wai-aria-roadmap-20080204/. Latest version of WAI-ARIA Roadmap available at http://www.w3.org/TR/wai-aria-roadmap/.

[ATAG1]
Authoring Tool Accessibility Guidelines 1.0, C. McCathieNevile, J. Treviranus, J. Richards, I. Jacobs, Editors, W3C Recommendation, 3 February 2000, http://www.w3.org/TR/2000/REC-ATAG10-20000203/. Latest version available at http://www.w3.org/TR/ATAG10/.

[ATK]
: *Gnome Accessibility Toolkit*. Available at http://library.gnome.org/devel/atk/unstable/.

[CSS]
: *Cascading Style Sheets, level 2 (CSS2) Specification*, I. Jacobs, B. Bos, H. Lie, C. Lilley, Editors, W3C Recommendation, 12 May 1998, http://www.w3.org/TR/1998/REC-CSS2-19980512/. Latest version of CSS2 available at http://www.w3.org/TR/REC-CSS2/.

[DOM]
: *Document Object Model (DOM) Level 2 Core Specification*, L. Wood, G. Nicol, A. Le Hors, J. Robie, S. Byrne, P. Le Hégaret, M. Champion, Editors, W3C Recommendation, 13 November 2000, http://www.w3.org/TR/2000/REC-DOM-Level-2-Core-20001113/. Latest version of DOM Core available at http://www.w3.org/TR/DOM-Level-2-Core/.

[IA2]
: *IAccessible2*. Available at http://www.linuxfoundation.org/en/Accessibility/IAccessible2.

[JAPI]
: *Java Accessibility API (JAPI)*. Available at http://java.sun.com/javase/technologies/accessibility/index.jsp.

[HTML5]
: *HTML 5*, D. Hyatt, I. Hickson, Editors, W3C Working Draft (work in progress), 22 January 2008, http://www.w3.org/TR/2008/WD-html5-20080122/. Latest version of HTML 5 available at http://www.w3.org/TR/html5/.

[MSAA]
: *Microsoft Active Accessibility (MSAA)*. Available at http://msdn.microsoft.com/en-us/library/ms697707.aspx.

[OWL]
: *OWL Web Ontology Language Overview*, D. L. McGuinness, F. van Harmelen, Editors, W3C Recommendation, 10 February 2004, http://www.w3.org/TR/2004/REC-owl-features-20040210/. Latest version of OWL Overview available at http://www.w3.org/TR/owl-features/.

[RDF]
: *Resource Description Framework (RDF): Concepts and Abstract Syntax*, G. Klyne, J. J. Carroll, Editors, W3C Recommendation, 10 February 2004, http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/. Latest version of RDF Concepts available at http://www.w3.org/TR/rdf-concepts/.

[RDFS]
: *RDF Vocabulary Description Language 1.0: RDF Schema*, D. Brickley, R. V. Guha, Editors, W3C Recommendation, 10 February 2004, http://www.w3.org/TR/2004/REC-rdf-schema-20040210/. Latest version of RDF Schema available at http://www.w3.org/TR/rdf-schema/.

[RFC2119]
: *Key words for use in RFCs to indicate requirement levels*, RFC 2119, S. Bradner, March 1997. Available at: http://www.rfc-editor.org/rfc/rfc2119.txt.

[SMIL]
: *Synchronized Multimedia Integration Language (SMIL) 1.0 Specification*, P. Hoschka, Editor, W3C Recommendation, 15 June 1998, http://www.w3.org/TR/1998/REC-smil-19980615/. Latest version of SMIL available at http://www.w3.org/TR/REC-smil/.

[SVG]

Scalable Vector Graphics (SVG) 1.1 Specification, D. Jackson, J. Ferraiolo, 藤沢, Editors, W3C Recommendation, 14 January 2003, http://www.w3.org/TR/2003/REC-SVG11-20030114/. Latest version of SVG available at http://www.w3.org/TR/SVG11/.

[UAAG]
User Agent Accessibility Guidelines 1.0, I. Jacobs, J. Gunderson, E. Hansen, Editors, W3C Recommendation, 17 December 2002, http://www.w3.org/TR/2002/REC-UAAG10-20021217/. Latest version available at http://www.w3.org/TR/UAAG10/.

[XFORMS]
XForms 1.0 (Third Edition), J. Boyer, Editor, W3C Recommendation, 29 October 2007, http://www.w3.org/TR/2007/REC-xforms-20071029/. Latest version of XForms available at http://www.w3.org/TR/xforms/.

[XHTML]
XHTML™ 1.0 The Extensible HyperText Markup Language (Second Edition), S. Pemberton, Editor, W3C Recommendation, 1 August 2002, http://www.w3.org/TR/2002/REC-xhtml1-20020801/. Latest version of XHTML 1.0 available at http://www.w3.org/TR/xhtml1/.

[XHTML2]
XHTML™ 2.0, M. Birbeck, J. Axelsson, S. Pemberton, B. Epperson, S. McCarron, M. Ishikawa, A. Navarro, M. Dubinko, Editors, W3C Working Draft (work in progress), 26 July 2006, http://www.w3.org/TR/2006/WD-xhtml2-20060726/. Latest version of XHTML 2.0 available at http://www.w3.org/TR/xhtml2/.

[XHTML-ROLES]
XHTML Role Attribute Module, T. V. Raman, M. Birbeck, R. Schwerdtfeger, S. McCarron, S. Pemberton, Editors, W3C Working Draft (work in progress), 4 October 2007, http://www.w3.org/TR/2007/WD-xhtml-role-20071004/. Latest version of XML Role Attribute Module available at http://www.w3.org/TR/xhtml-role/.

[XML]
Extensible Markup Language (XML) 1.0 (Fourth Edition), T. Bray, J. Paoli, E. Maler, C. M. Sperberg-McQueen, F. Yergeau, Editors, W3C Recommendation, 16 August 2006, http://www.w3.org/TR/2006/REC-xml-20060816/. Latest version of XML available at http://www.w3.org/TR/xml/.

[XML-EVENTS]
XML Events 2, M. Birbeck, S. McCarron, Editors, W3C Working Draft (work in progress), 16 February 2007, http://www.w3.org/TR/2007/WD-xml-events-20070216/. Latest version of XML Events available at http://www.w3.org/TR/xml-events/.

[XML-NAMES]
Namespaces in XML 1.0 (Second Edition), D. Hollander, A. Layman, R. Tobin, T. Bray, Editors, W3C Recommendation, 16 August 2006, http://www.w3.org/TR/2006/REC-xml-names-20060816/. Latest version of XML Namespaces available at http://www.w3.org/TR/xml-names/.

[XSD]
XML Schema Part 0: Primer Second Edition, D. C. Fallside, P. Walmsley, Editors, W3C Recommendation, 28 October 2004, http://www.w3.org/TR/2004/REC-xmlschema-0-20041028/. Latest version of XML Schema Primer available at http://www.w3.org/TR/xmlschema-0/.

7.4 Acknowledgments

This section is *informative*.

The following contributed to the development of this document.

7.4.1 Participants in the PFWG at the time of publication

- Jim Allan (Invited Expert, Texas School for the Blind)
- Chris Blouch (AOL)
- David Bolter (Invited Expert, University of Toronto Adaptive Technology Resource Centre)
- Sally Cain (Royal National Institute for the Blind)
- Charles Chen (Google)
- Michael Cooper (W3C/MIT)
- James Craig (Apple, Inc.)
- Dimitar Denev (Frauenhofer Gesellschaft)
- Donald Evans (AOL)
- Steve Faulkner (Invited Expert, The Paciello Group)
- Kentarou Fukuda (IBM Corporation)
- Alfred S. Gilman (Invited Expert)
- Andres Gonzalez (Adobe Systems Inc.)
- Georgios Grigoriadis (SAP AG)
- Jon Gunderson (Invited Expert, UIUC)
- Sean Hayes (Microsoft Corporation)
- John Hrvatin (Microsoft Corporation)
- Kenny Johar (Vision Australia)
- Masahiko Kaneko (Microsoft Corporation)
- Diego La Monica (IWA/HWG)
- Aaron Leventhal (IBM Corporation)
- Alex Li (SAP AG)
- William Loughborough (Invited Expert)
- Anders Markussen (Opera Software)
- Matthew May (Adobe)
- Charles McCathieNevile (Opera Software)
- James Nurthen (Oracle Corporation)
- Joshue O'Connor (Invited Expert)
- Lisa Pappas (Invited Expert, SAS)
- Simon Pieters (Opera)
- David Poehlman (Invited Expert)
- T.V. Raman (Google, Inc.)
- Gregory Rosmaita (Invited Expert)
- Janina Sajka (Invited Expert, The Linux Foundation)
- Martin Schaus (SAP AG)
- Joseph Scheuhammer (Invited Expert, University of Toronto Adaptive Technology Resource Centre)
- Stefan Schnabel (SAP AG)
- Richard Schwerdtfeger (IBM Corporation)
- Lisa Seeman (Invited Expert, Aqueous)
- Cynthia Shelly (Microsoft Corporation)
- Marc Silbey (Microsoft Corporation)
- Gottfried Zimmermann (Invited Expert, Access Technologies Group)

7.4.2 Other previously active PFWG participants and other contributors to the Accessible Rich Internet Applications specification

Special thanks to Aaron Leventhal for effort and insight as he implemented a working prototype of accessibility API bindings.

Simon Bates, Judy Brewer (W3C/MIT), Christian Cohrs, Becky Gibson (IBM), Andres Gonzalez (Adobe), Jeff Grimes (Oracle), Barbara Hartel, Earl Johnson (Sun), Jael Kurz, Linda Mao (Microsoft), Shane McCarron (ApTest), Dave Pawson (RNIB), Henri Sivonen (Mozilla), Vitaly Sourikov, Mike Squillace (IBM), Ryan Williams (Oracle), Tom Wlodkowski.

7.4.3 Enabling funders

This publication has been funded in part with Federal funds from the U.S. Department of Education under contract number ED05CO0039. 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.