WAI-ARIA Graphics Module

W3C Recommendation 02 October 2018

This version:
https://www.w3.org/TR/2018/REC-graphics-aria-1.0-20181002/

Latest published version:
https://www.w3.org/TR/graphics-aria-1.0/

Latest editor's draft:
https://w3c.github.io/graphics-aria/

Implementation report:
https://w3c.github.io/test-results/graphics-aam/

Previous version:
https://www.w3.org/TR/2018/PR-graphics-aria-1.0-20180626/

Editors:
Amelia Bellamy-Royds
Joanmarie Diggs (Igalia, S.L.)
Michael Cooper (W3C)

Former editors:
Fred Esch (IBM Corporation) (until September 2016)
Rich Schwerdtfeger (Knowbility) (until August 2017)

Authors:
Amelia Bellamy-Royds
Fred Esch (IBM Corporation)
Rich Schwerdtfeger (Knowbility)
Doug Schepers (W3C)

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

See also **translations**.

Abstract

Assistive technologies need semantic information about the structures and expected behaviors of a document in order to convey appropriate information to persons with disabilities. This specification defines a WAI-ARIA 1.1 [WAI-ARIA-1.1] module of core roles specific to web graphics. These semantics allow an author to express the logical structure of the graphic to assistive technologies in order improve accessibility of graphics. Assistive technologies could then enable semantic navigation and adapt styling and interactive features, to provide an optimal experience for the audience. These

features complement the graphics and document structure elements defined by HTML [HTML52] and SVG [SVG2].

This document is part of the WAI-ARIA suite described in the WAI-ARIA Overview.

Status of This Document

This section describes the status of this document at the time of its publication. Other documents may supersede this document. A list of current W3C publications and the latest revision of this technical report can be found in the W3C technical reports index at https://www.w3.org/TR/.

This is the Graphics-ARIA 1.0 W3C Recommendation by the Accessible Rich Internet Applications Working Group. The Working Group created a Graphics-ARIA 1.0 Implementation Report to demonstrate that the specification is implementable. A history of changes to Graphics-ARIA 1.0 is available in the appendix.

To comment, file an issue in the W3C graphics-aria GitHub repository. If this is not feasible, send email to public-aria@w3.org (comment archive). Comments received on the Graphics-ARIA 1.0 Recommendation cannot result in changes to this version of the specification, but may be addressed in errata or future versions of Graphics-ARIA. The Working Group may not make formal responses to comments but future work undertaken by the Working Group may address comments received on this document. In-progress updates to the document may be viewed in the publicly visible editors' draft.

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

Comments regarding this document are welcome. Please send them to public-aria@w3.org (archives).

Please see the Working Group's implementation report.

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

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

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

Table of Contents

1. Introduction §

This section is non-normative.

WAI-ARIA is a technical specification that provides a framework to improve the accessibility and interoperability of web content and applications. It enables web browsers to map the accessibility

semantics in web content to platform-specific accessibility APIs. This enables web content to be interoperable with platform assistive technologies, similar to native platform applications, without requiring authors to include platform dependencies.

This specification is a modular extension of WAI-ARIA [WAI-ARIA-1.1] designed to support graphics. The goals of this specification include:

- Expanding WAI-ARIA to produce semantic extensions to support structured graphics such as charts, graphs, maps, technical drawings and scientific diagrams. It has applicability to both Scalable Vector Graphics as well as HTML 5 Canvas and graphics produced with CSS styling of HTML and other markup languages.
- Align with a new governance model for modularization and extensions to WAI-ARIA.
- Provide structural semantics extensions that will support both assistive technologies and enable semantic navigation, alternative styling, and interactivity support.
- Work in harmony with SVG 2 and ARIA 1.1 to get consistent working accessibility infrastructure, on par with WAI-ARIA and HTML 5.2, across all the major browsers.

This specification defines the core roles that would be used in all structured graphics or diagrams. It establishes the default roles that can be used to describe graphical markup elements such as shapes and canvases. In combination with WAI-ARIA attributes to provide alternative text and to indicate relationships between elements, this provides a framework for annotating many figures and diagrams. Future work will expand on this framework to enable more detailed annotation of data-rich graphics such as charts or maps.

For a more detailed explanation of WAI-ARIA please refer to the WAI-ARIA Introduction and how it applies to Rich Internet Application Accessibility.

1.1 Target Audience §

This specification defines a module of WAI-ARIA for graphics, consisting of graphics-specific element roles. It impacts several audiences:

- User agents that process content containing WAI-ARIA and graphics WAI-ARIA features;
- Assistive technologies that provide specialized reading experiences to users with disabilities;
- Authors of web graphics;
- Authoring tools that help authors create conforming graphics; and
- Conformance checkers, that verify appropriate use of WAI-ARIA and this WAI-ARIA Graphics module.

Each conformance requirement indicates the audience to which it applies.

1.2 User Agent Support §

This module follows the general User Agent support principles defined in WAI-ARIA [WAI-ARIA-1.1]. The roles defined here do not require any change in behavior by user agents other than in the information exposed to the accessibility API. However, the semantics defined here provide the ability for user agents to enhance the general user interface presented to readers. For example, a user agent may provide alternative keyboard navigation suitable to a graphical environment, or may allow users to extract a copy of a graphic from a larger document.

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

The WAI-ARIA Graphics module follows the model for co-evolution of WAI-ARIA and host languages defined in WAI-ARIA [WAI-ARIA-1.1]. It is intended to augment semantics in supporting languages like HTML [HTML52], SVG [SVG2] and EPUB, or to be used as an accessibility enhancement technology in other markup-based languages that do not explicitly include support for ARIA. WAI-ARIA roles clarify semantics to assistive technologies when authors create new types of objects, via style and script, or use markup languages which describe the visual appearance of a document rather than its meaning.

Although markup languages may provide some of these semantics natively, it is expected that there will be a persistent need for the semantics provided by the WAI-ARIA Graphics module. Some host languages exist to create semantics for features other than the user interface. For example, SVG expresses the semantics behind production of graphical objects, not of user interface components that those objects may represent. Host languages such as these might, by design, not provide native semantics that map to all of this specification's features. In these host languages, the WAI-ARIA Graphics module could be adopted as a long-term approach to add semantic information.

1.4 Authoring Practices §

1.4.1 Authoring Tools §

Many of the requirements in the definitions of the WAI-ARIA and Graphics WAI-ARIA roles, states and properties can be checked automatically during the development process, similar to other quality control processes used for validating code. To assist authors who are creating graphics, these tools can compare the semantic structure of Graphics WAI-ARIA roles from the DOM to that defined in this specification and notify the author of errors or simply create templates that enforce that structure.

1.4.2 Testing Practices and Tools §

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

1.5 Assistive Technologies §

Programmatic access to accessibility semantics is essential for assistive technologies. For more information, refer to the Assistive Technologies section in WAI-ARIA [WAI-ARIA-1.1].

For the graphics roles in particular, two categories of assistive technology are particularly relevant, but have different needs:

- Text-based presentations, such as screen readers, braille displays, and text-only displays or printers. These technologies need to replace a complex graphic with semantic text descriptions, preserving any meaningful structure and relationships between components.
- Alternative graphical presentations, such as colour-adjusted displays, screen magnifiers, large print documents, or embossing printers with graphic support. These technologies need to distinguish between graphical features which are primarily decorative and those which are essential for conveying the meaning of the content.

The role descriptions suggest which features of an element with that role are considered semantically important and should be conveyed to the reader whenever possible.

2. Conformance §

The main content of this specification is normative and defines requirements that impact conformance claims. Introductory material, appendices, sections marked as "non-normative" and their subsections, diagrams, examples, and notes are informative (non-normative). Non-normative material provides advisory information to help interpret the guidelines but does not create requirements that impact a conformance claim.

Normative sections provide requirements that *user agents* must follow 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 *Keywords for use in RFCs to indicate requirement levels* [RFC2119]. RFC-2119 keywords are formatted in uppercase and contained in an element with `class="rfc2119"`. When the keywords shown above are used, but do not share this format, they do not convey formal information in the RFC 2119 sense, and are merely explanatory, i.e., informative. As much as possible, such usages are avoided in this specification.

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

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

3. Important Terms §

This section is non-normative.

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

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

Assistive Technologies
Hardware and/or software that:

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

This definition may differ from that used in other documents.

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

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

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

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

Element
In this specification, element is used as it is in markup languages. Elements are the structural

elements in markup language that contains the data profile for *objects*.

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

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

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

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

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

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

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

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

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

State
A state is a dynamic *property* expressing characteristics of an object that may change in response

to user action or automated processes. States do not affect the essential nature of the object, but represent data associated with the object or user interaction possibilities. See clarification of states versus properties.

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

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

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

4. Graphics Roles §

This section defines additions to the WAI-ARIA role taxonomy and describes the characteristics and properties of all roles. See ARIA Roles for descriptions of the fields provided by this module.

Authors are given the ability to influence what is presented to assistive technologies and to influence navigation through the use of roles and properties. This includes the ability to mark elements as having no semantic importance. With graphics, there are many cases where presenting and navigating every element will make the graphic harder to understand and use.

Authors may mark elements for exclusion from the semantic representation of the document (the accessibility tree) by assigning the role `none` or `presentation`. The element with this role should be treated transparently by assistive technologies, as if its children or text content were directly contained by its parent element. In addition, certain roles, such as `img` or `graphics-symbol`, when assigned to a parent element, will cause all child DOM structure to be omitted from the accessibility tree. This is indicated by the "Children Presentational" value in the role characteristics table. Finally, the native semantics of the graphics language may also default to ignoring DOM structure that does not have semantic data attached; for SVG, this is defined in the SVG Accessibility API Mappings specification [SVG-AAM-1.0].

In all cases, to be considered presentational, an element must not be interactive and must not be assigned any accessible properties or alternative text. A role of `none` or `presentation` will be ignored for interactive elements or those with WAI-ARIA states and properties.

4.1 Definition of Roles §

Below is an alphabetical list of the WAI-ARIA roles defined in this specification. They would normally be used in combination with other roles defined in WAI-ARIA to annotate graphics within

documents and rich internet applications [WAI-ARIA-1.1].

graphics-document
: A type of `document` in which the visual appearance or layout of content conveys meaning.

graphics-object
: A section of a `graphics-document` that represents a distinct object or sub-component with semantic meaning. A graphical object may itself have nested sub-components.

graphics-symbol
: A graphical object used to convey a simple meaning or category, where the meaning is more important than the particular visual appearance. It may be a component of a larger structured graphic such as a chart or map. The symbol itself is an atomic object; children are presentational.

`graphics-document` (role) §

A type of `document` in which the visual appearance or layout of content conveys meaning.

Similar to other `document` types, the `graphics-document` role applies to the root element of a region of the page containing related information, where the user's primary interaction mode is expected to be browsing the document rather than controlling an application. The element with this role may be the root element of the document file, or of a nested structure within it.

The `graphics-document` may be distinguished from similar roles as follows:

- Relative to other documents, a `graphics-document` is distinguished by the semantic importance of its visual (usually two-dimensional) representation. User agents and assistive technologies ***SHOULD*** take this into consideration when supporting navigation of the graphic. Accessibility technologies that re-format or re-style a document ***SHOULD NOT*** alter the layout of a `graphics-document` except in ways that are consistent with the semantic roles and relationships of its content.

- Relative to an `img`, a `graphics-document` is distinguished by the structured nature of its content. Its child elements may have semantic meaning, and may include links or other interactive widgets.

- Relative to a `graphics-object`, a `graphics-document` is self-contained. Its meaning persists when separated from surrounding content. The element with the `graphics-document` role defines the scope and context for interpretation of the child content.

In general, authors ***SHOULD*** use the `graphics-document` role for structured graphics such as charts, maps, diagrams, technical drawing, blue prints and instructional graphics. However, if a single large graphic has discrete regions that may be safely re-arranged without sacrificing meaning, each of those regions ***SHOULD*** be a distinct `graphics-document`. An alternative role (such as `figure`) may be used to group them together. One `graphics-document` may also be nested inside another, for example a bar chart that is embedded in a map or a matrix of chart panels should have a role of `graphics-document`. The nested document provides encapsulation; navigation between components of the inner and outer graphics should be explicit.

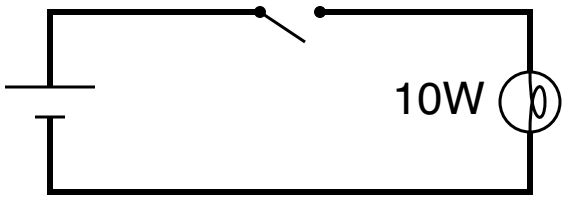
NOTE

To support user agents and assistive technologies based on the ARIA 1.0 specification, authors may wish to include the `document` role as a fallback value, in the form `role="graphics-document document"`.

Future specifications may define more specific roles for particular types of graphical documents with special semantic structures. Those more specific roles would be subclasses of `graphics-document`.

EXAMPLE 1

An SVG diagram of an electrical circuit is a simple graphical document:




```
<svg xmlns="https://www.w3.org/2000/svg"
     width="400" height="200" viewBox="0 0 200 100"
     role="graphics-document document" >
     <title>A simple circuit</title>
     <desc>A circuit with one source, one switch, and one load</desc>
     <style type="text/css">
        /* omitted */
     </style>
     <g id="battery-1" role="graphics-symbol img"
        aria-roledescription="source" aria-label="battery"
        transform="translate(20,50)">
        <path d="M-15,-5 h30 M-5,5 h10"/>
     </g>
     <path id="wire-1" role="graphics-symbol img"
          aria-label="wire connecting"
          aria-labelledby="wire-1 battery-1 switch-1"
          class="wire" d="M20,45 V20 H90"/>
     <!-- the switch, more wires, and a light bulb -->
</svg>
```

Characteristics:

Characteristic	Value
Superclass Role:	document

Characteristic	Value
Related Concepts:	graphics-object img article
Inherited States and Properties:	aria-atomic aria-busy (state) aria-controls aria-current (state) aria-describedby aria-details aria-disabled (state) aria-dropeffect aria-errormessage aria-expanded (state) aria-flowto aria-grabbed (state) aria-haspopup aria-hidden (state) aria-invalid (state) aria-keyshortcuts aria-label aria-labelledby aria-live aria-owns aria-relevant aria-roledescription
Name From:	author
Accessible Name Required:	True
Children Presentational:	False

graphics-object (role) §

A section of a graphics-document that represents a distinct object or sub-component with semantic meaning. A graphical object may itself have nested sub-components.

Container elements that represent a collection of disconnected objects should be given the `group` or `list` roles, instead. Grouping elements that do not have semantic meaning and do not alter the semantic context provided by an ancestor (for example, a `div` or SVG `g` that is only used for styling or layout) ***SHOULD NOT*** be given a role. The lack of role may be explicitly indicated with the role `none` or `presentation`.

Unlike a `graphics-document`, a `graphics-object` need not be self-contained, and it does not establish a new context for navigation. However, user agents and assistive technologies ***SHOULD*** provide a way for users, particularly non-visual users, to navigate the nested structure of objects in a hierarchical manner, similar to nested lists.

> NOTE
>
> To support user agents and assistive technologies based on the ARIA 1.0 specification, authors may wish to include the `group` role as a fallback value, in the form `role="graphics-object group"`.

EXAMPLE 2

The code that follows is a portion of the markup for a structured graphic. It includes SVG `g` grouping elements with various roles:

- `graphics-object` for distinct objects, such as the house, its door, or roof.
- `group` to group together the windows or the trees (multiple distinct objects) with a single label or description.
- `img` for the background which is described as a whole.
- `none` for elements that apply styles or transformations without having any semantic meaning.

Where a graphical object has multiple sub-components, the group role is provided as an explicit fallback.


```
<svg xmlns="https://www.w3.org/2000/svg"
     width="600" height="400" viewBox="0 0 600 400"
     role="graphics-document document" xml:lang="en">
    <title>Home</title>
    <g role="img" aria-label="background">
        <desc>Blue sky, sunshine, and green grass</desc>
        <!-- The multiple parts of the background form a single image
             conveyed by that one description. -->
        <rect fill="lightSkyBlue" height="100%" width="100%" />
        <circle fill="yellow" stroke="gold" stroke-width="4"
                cx="0" cy="0" r="50" />
        <path fill="none" stroke="gold" stroke-width="3"
              d="..." />
        <rect fill="#6a2" y="300" width="100%" height="100" />
    </g>
    <g role="graphics-object group"
       aria-labelledby="house-label"
       transform="translate(100,325)">
        <desc>A two-storey brick house, drawn with basic shapes.
        </desc>
        <!-- The house has a number of details worth calling out,
             so it is a graphical object -->

        <rect fill="firebrick" stroke="darkRed"
              width="300" height="200" y="-200"
              role="none" /><!-- the walls of the house are
                already described thoroughly,
                so no role is required -->
```

```
        <g role="graphics-object" aria-label="door"
           transform="translate(30,-90)">
            <desc>The brown door on the left side of the building
                has a window and a round doorknob</desc>
            <!-- The graphical object role allows for further
                 nested sub-components.
                 However, based on the default SVG API mappings,
                 these shapes, which have neither labels nor descriptions,
                 will be treated as presentation. -->
            <rect fill="darkKhaki" stroke="#632"
                  width="50" height="90"/>
            <rect fill="lightSteelBlue" stroke="#632" stroke-width="4"
                  x="5" y="5" width="40" height="30" />
            <circle fill="gray" stroke="#444" stroke-width="0.7"
                    cx="10" cy="50" r="4" />
        </g>

        <g role="group" aria-label="windows"
           fill="lightSteelBlue" stroke="#632" stroke-width="6">
            <!-- The windows are distinct objects,
                 grouped together with a common label -->
            <g role="none" transform="translate(0,-85)">
                <rect aria-label="first-floor window"
                      x="100" width="25" height="45">
                    <desc>A small window beside the door</desc>
                </rect>
                <path aria-label="first-floor living-room window"
                      d="M180,0h100v60h-100v-60z
                         m30,0v60 m40,0v-60">
                    <desc>A large three-pane window fills
                    the rest of the first floor</desc>
                </path>
            </g>
            <g role="none" transform="translate(0,-180)">
                <!-- more windows on the second floor -->
            </g>
        </g>
        <!-- more markup for the roof and chimney -->

        <text id="house-label"
              font-family="cursive" font-size="36px"
              x="70" y="50">My House</text>
    </g>
    <!-- more markup for the trees -->
</svg>
```

Characteristics:

Characteristic	Value
Superclass Role:	group

Characteristic	Value
Related Concepts:	graphics-document group img graphics-symbol
Inherited States and Properties:	aria-activedescendant aria-atomic aria-busy (state) aria-controls aria-current (state) aria-describedby aria-details aria-disabled (state) aria-dropeffect aria-errormessage aria-expanded (state) aria-flowto aria-grabbed (state) aria-haspopup aria-hidden (state) aria-invalid (state) aria-keyshortcuts aria-label aria-labelledby aria-live aria-owns aria-relevant aria-roledescription
Name From:	author contents
Accessible Name Required:	False
Children Presentational:	False

`graphics-symbol` (role) §

A graphical object used to convey a simple meaning or category, where the meaning is more important than the particular visual appearance. It may be a component of a larger structured graphic such as a chart or map. The symbol itself is an atomic object; children are presentational.

When used as part of a structured symbolic language, the `aria-roledescription` property (introduced in ARIA 1.1 [WAI-ARIA-1.1]) can be used to name the symbol type separately from the name and description for the particular instance of the symbol.

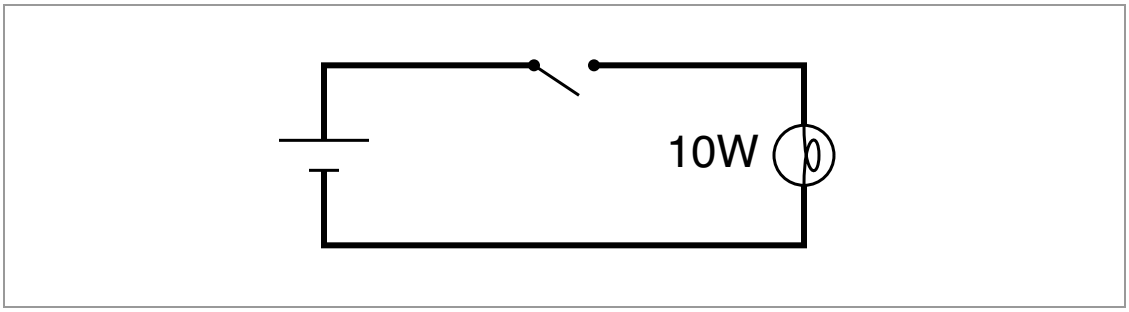
> NOTE
>
> To support user agents and assistive technologies based on the ARIA 1.0 specification, authors may wish to include the `img` role as a fallback value, in the form `role="graphics-symbol img"`, if that is not already the default semantic role for the element.

> EXAMPLE 3
>
> Within an HTML document for a restaurant menu, an `img` element can represent a repeated symbol:

```
<h2>Appetizers</h2>
<ul>
    <li> Spinach Salad with Strawberry & Almonds
        <img role="graphics-symbol" title="Vegetarian" src="../assets/leaf.svg" />
        <img role="graphics-symbol" title="Contains Nuts" src="../assets/peanut.svg" />
    </li>
    <li> Chicken Satay with Peanut Sauce
        <img role="graphics-symbol" title="Contains Nuts" src="../assets/peanut.svg" />
    </li>
    <!-- … -->
</ul>
```

EXAMPLE 4

Within an SVG diagram of an electrical circuit, the graphics that represent batteries, switches, and loads like this lightbulb are each symbols:

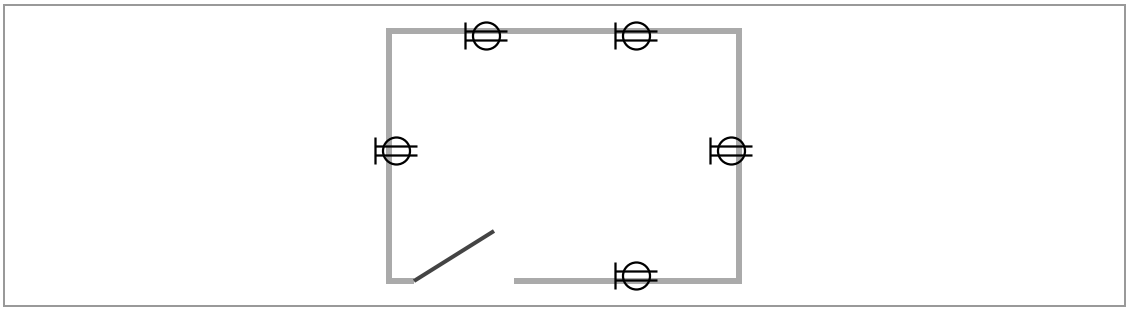



```
<g id="lightbulb-1" role="graphics-symbol img"
   aria-roledescription="load"
   aria-label="lightbulb"
   aria-labelledby="lightbulb-1 lightbulb-1-label"
   transform="translate(180,50)">
    <text id="lightbulb-1-label"
        x="-15" dy="0.5ex" text-anchor="end" >10W</text>
    <circle r="10" />
    <path d="M0,-10 C0,8 5,8 5,0 C5,-8 0,-8 0,10" />
</g>
```

Note that the visible text must be included in the label for its parent symbol, in this example. As the child of a `graphics-symbol`, it is treated as presentational content, and is therefore not accessible as a separate element of the graphic.

EXAMPLE 5

Within an architectural blueprint-style SVG diagram, each SVG `use` element that creates a copy of a simple SVG `symbol` is a `graphics-symbol`:



```
<svg xmlns="http://www.w3.org/2000/svg"
     xmlns:xlink="http://www.w3.org/1999/xlink"
     width="400" height="300" viewBox="0 0 400 300"
     role="graphics-document document" xml:lang="en">
    <title>Room with 5 outlets</title>
    <desc>Schematic showing the minimum number and position of
        electrical outlets for a mid-sized room with one door.
    </desc>
    <defs>
        <symbol id="outlet" viewBox="0 0 30 20"
                stroke="#000" stroke-width="1.5" pointer-events="all">
          <desc>The symbol for an electrical outlet is a circle with
          a plug shape overlaid on top.  The plug consists of two horizontal
          lines extending from a vertical line.</desc>
          <circle cx="15" cy="10" r="9" fill="none"/>
          <line x1="1" y1="7" x2="29" y2="7"/>
          <line x1="1" y1="13" x2="29" y2="13"/>
          <line x1="1" y1="1" x2="1" y2="19"/>
        </symbol>
    </defs>
<!-- … -->
    <g role="group">
        <use xlink:href="#outlet" role="graphics-symbol img"
             x="100" y="15" width="45" height="30">
            <title>Electrical outlet</title>
            <desc>on West side of North wall</desc>
        </use>
        <use xlink:href="#outlet" role="graphics-symbol img"
             x="250" y="15" width="45" height="30">
            <title>Electrical outlet</title>
            <desc>on East side of North wall</desc>
        </use>
<!-- … -->
    </g>
</svg>
```

Characteristics:

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

4.2 Other Roles for Graphics §

The following core ARIA roles, defined in ARIA 1.1 [WAI-ARIA-1.1], are also relevant for annotating graphics:

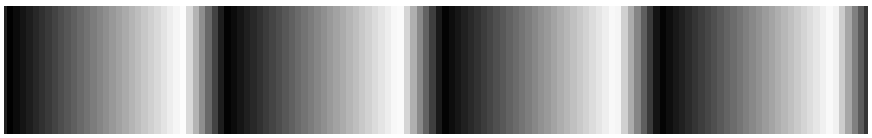
- `img` (image) defines a single graphic that is perceived as an indivisible whole. Unlike a `graphics-document`, an image cannot have navigable or interactive child content. Unlike a `graphics-symbol`, an image may require a detailed text description to fully convey its meaning to non-visual users.
- `figure` defines a container element for content (including graphics) that is a key part of the containing document but is outside the normal reading stream. A figure will often contain one or more elements with the `img` or `graphics-document` roles, but may also contain text captions, credits, or other related content.

The following examples demonstrate appropriate use of `img`, `figure`, and `graphics-document` in a document.

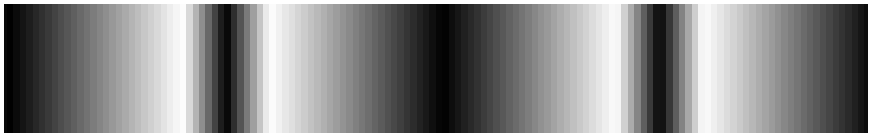
EXAMPLE 6

Within an HTML 5 document, an inline SVG may sometimes represent an atomic `img`. If the graphics form part of the natural reading flow of the text, then this role is sufficient, as in the first example:

A repeating SVG gradient is defined using the `spreadMethod` attribute. A value of `repeat` causes the color stops to repeat in the same order, from beginning to end:



In contrast, a value of `reflect` causes the order and spacing of colors to alternate in each repeat:



Both of these gradients repeat the sequence of color stops four times, from

```
<p>A repeating SVG gradient is defined using the
    <code>spreadMethod</code> attribute.
    A value of <code>repeat</code> causes the color stops
    to repeat in the same order, from beginning to end:</p>
<svg class="inline-example" role="img">
    <title>A repeating linear gradient</title>
    <desc>The gradient starts dark, slowly shifting to light,
        then quickly dark again.  This pattern repeats four
        times left to right, each time brightening across a large
        region and then getting dark within a short space.
    </desc>
    <linearGradient id="repeat" x2="25%" spreadMethod="repeat">
        <stop offset="0"     stop-color="black" />
        <stop offset="0.8" stop-color="white" />
        <stop offset="1"     stop-color="black" />
    </linearGradient>
    <rect width="100%" height="100%" fill="url(#repeat)" />
</svg>
```

The following example provides the same content, but now structured as a figure that may be re-positioned separately from the flow of paragraph text.

a) spreadMethod="repeat"

b) spreadMethod="reflect"

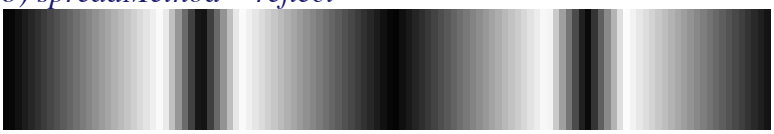


Figure 1: Repeating SVG gradients

A repeating SVG gradient is defined using the `spreadMethod` attribute. A value of `repeat` causes the color stops to repeat in the same order, from beginning to end, as shown in Figure 1-a.

```
<figure id="fig1" role="figure region">
    <svg id="fig1A" class="nested-figure" role="figure"
         aria-labelledby="fig1-caption, fig1A-caption">
        <!-- markup omitted-->
    </svg>
    <svg id="fig1B" class="nested-figure" role="figure"
         aria-labelledby="fig1-caption, fig1B-caption">
        <linearGradient id="reflect" spreadMethod="reflect"
                        xlink:href="#repeat" />
        <text id="fig1B-caption" class="caption" dy="1em"
              >b) spreadMethod="reflect"</text>
        <rect role="img"
              y="25%" width="100%" height="75%" fill="url(#reflect)" >
            <title>A reflecting linear gradient</title>
            <desc>The gradient again starts dark, slowly shifting to light,
              then quickly dark again.  However, the next repeat shifts
              quickly to the light, then slowly back dark.
              The original pattern is then repeated, followed by the reflected
              version again.
            </desc>
        </rect>
    </svg>
    <figcaption id="fig1-caption"
                >Figure 1: Repeating SVG gradients</figcaption>
</figure>
```

Finally, the last version uses a `graphics-document` to include complex annotations on the graphic, which is still contained within a figure element. The two sections of the graphic are `graphics-object` elements, while the annotations have individual labels and descriptions.

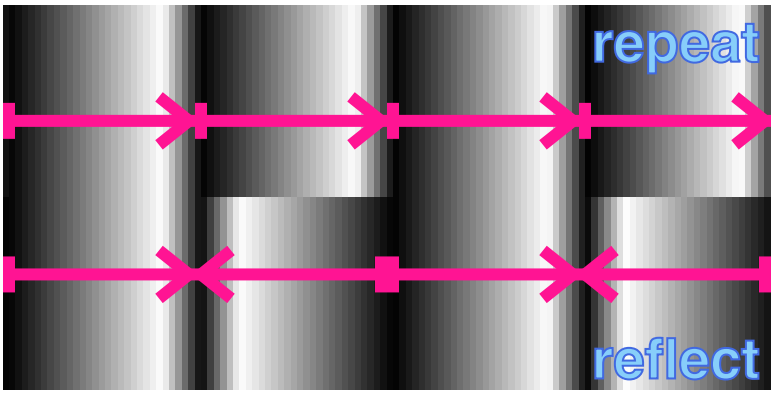


Figure 1: spreadMethod options for repeating SVG gradients

A repeating SVG gradient is defined using the `spreadMethod` attribute.

```
<figure id="fig1" role="figure region">
    <svg role="graphics-document">
        <title>Repeating versus reflecting linear gradients</title>
        <desc>
            The graphic shows two gradient patterns,
            annotated with text labels and arrows.
            Both gradients use the same series of color stops,
            starting dark, slowly shifting to light,
            then quickly dark again.
            This cycle covers one-quarter of the width of the graphic,
            starting on the left.
            The two gradients differ in their repeat cycles.
        </desc>
        <defs>
            <!-- markup omitted -->
        </defs>
        <g role="graphics-object" aria-labelledby="repeat-label">
            <!-- markup omitted -->
        </g>
        <g role="graphics-object" aria-labelledby="reflect-label">
            <desc>
                The gradient stretches across the bottom half of
                the graphic.
                Each cycle of the gradient alternates direction,
                left-to-right then right-to-left,
                as do the arrows that emphasize the pattern.
            </desc>
            <rect y="50%" width="100%" height="50%" fill="url(#reflect)" />
            </rect>
            <use y="70%" xlink:href="#gradient-vector" >
                <title>1st cycle</title>
                <desc>left-to-right arrow, starting from x="0"</desc>
            </use>
            <use y="70%" x="-50%" transform="scale(-1,1)"
                 xlink:href="#gradient-vector" >
```

```
                <title>2nd cycle, reflected</title>
                <desc>right-to-left arrow, ending at x="25%"</desc>
            </use>
            <use y="70%" x="50%" xlink:href="#gradient-vector" >
                <title>3rd cycle</title>
                <desc>left-to-right arrow, starting from x="50%"</desc>
            </use>
            <use y="70%" x="-100%" transform="scale(-1,1)"
                 xlink:href="#gradient-vector" >
                <title>4th cycle, reflected</title>
                <desc>right-to-left arrow, ending at x="75%"</desc>
            </use>
            <text id="reflect-label" class="overlay-text"
                  aria-label="spreadMethod='reflect'"
                  dy="-0.2em" dx="-0.2em" x="100%" y="100%"
                  >reflect</text>
        </g>
    </svg>
    <figcaption>Figure 1: spreadMethod options for
        repeating SVG gradients</figcaption>
</figure>
```

EXAMPLE 7

Within the fallback DOM dynamically constructed for an HTML 5 canvas game, elements may represent the different images that create the composite graphic:

```
<canvas role="graphics-document">
    <p id="scene-desc" role="img"
       aria-label="The Dungeon"
       aria-describedby="scene-desc">
       The door opens into a long hallway.
       Every few steps on either side are barred doors.
       Moisture drips from the stone walls.
    </p>
    <p id="grue-1" role="img"
       aria-label="Grue"
       aria-describedby="grue-1">
       An amorphous and poorly defined, but unmistakably sinister,
       creature blocks the passage.
   </p>
   <!-- more DOM elements representing game controls -->
</canvas>
```

5. States and Properties §

WAI-ARIA provides a collection of accessibility state and properties which are used to support

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

A. Change Log §

The full commit history to WAI-ARIA Graphics Module 1.0 is available.

A.1 Substantive changes since the last public working draft §

- 2017-11-15: Change superclass for graphics-object to img and graphics-symbol to group.

A.2 Other substantive changes since the First Public Working Draft §

- 2016-01-26: changed superclass of graphics-doc to structure.
- 2016-04-28: removed reference to role="none" or "presentation".
- 2016-05-02: Change graphics-doc to graphics-document.
- 2016-05-07: clarify limitations of presentational role; mention SVG 2 next to HTML 5 in section on Schemata.

B. Acknowledgments §

This section is non-normative.

The following people contributed to the development of this document.

B.1 Participants active in the SVG accessibility task force at the time of publication §

- Amelia Bellamy-Royds (Invited expert)
- Fred Esch (IBM Corporation)
- Charles McCathieNevile (Yandex)
- Charu Pandhi (IBM Corporation)
- Doug Schepers (W3C Staff)

- Richard Schwerdtfeger (Knowbility)
- Léonie Watson (The Paciello Group)
- Jason White (Educational Testing Service)

B.2 Participants active in the ARIA WG at the time of publication §

- David Bolter (Mozilla Foundation)
- Michael Cooper (W3C/MIT)
- James Craig (Apple Inc.)
- Joanmarie Diggs (Igalia)
- John Foliot (Invited Expert)
- Christopher Gallelo (Microsoft Corporation)
- Bryan Garaventa (SSB BART Group)
- Jon Gunderson (University of Illinois at Urbana-Champaign)
- Matthew King (IBM Corporation)
- Dominic Mazzoni (Google, Inc.)
- Shane McCarron (Invited Expert, Aptest)
- James Nurthen (Oracle Corporation)
- Janina Sajka (Invited Expert, The Linux Foundation)
- Stefan Schnabel (SAP AG)
- Lisa Seeman (Invited Expert)
- Alexander Surkov (Mozilla Foundation)
- Jason White (Educational Testing Service)

B.3 Enabling funders §

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

C. References §

C.1 Normative references §

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

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

C.2 Informative references §

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

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

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

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

[HTML52]
HTML 5.2. Steve Faulkner; Arron Eicholz; Travis Leithead; Alex Danilo; Sangwhan Moon. W3C. 14 December 2017. W3C Recommendation. URL: https://www.w3.org/TR/html52/

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

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

[SVG-AAM-1.0]
SVG Accessibility API Mappings. Amelia Bellamy-Royds; Ian Pouncey. W3C. 10 May 2018. W3C Working Draft. URL: https://www.w3.org/TR/svg-aam-1.0/

[SVG2]
Scalable Vector Graphics (SVG) 2. Amelia Bellamy-Royds; Bogdan Brinza; Chris Lilley; Dirk Schulze; David Storey; Eric Willigers. W3C. 7 August 2018. W3C Candidate Recommendation.

URL: https://www.w3.org/TR/SVG2/

[UI-AUTOMATION]

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

[UIA-EXPRESS]

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

[WCAG21]

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

↥