Review O: Accessibility

Table of contents

1 Accessibility

DITA has markup and features that enable producing output that is accessible by all audiences.

1.1 Handling accessibility in content and in processors

Accessibility requirements vary depending on how content is rendered. Making content accessible is work that involves both content authors and the processors that render DITA content.

The foundation for accessible content is the Web Content Accessibility Guidelines (WCAG) from W3C. While content formats and content authors might have unique or additional accessibility needs, the rules outlined in the WCAG provide a reference point for considering how to create accessible content in DITA.

The guidelines fall into several categories:

General content guidelines

Many accessibility guidelines and best practices apply to all content. Such guidelines are generally outside the scope of this specification.

For example, a guideline might recommend against multiple levels of nested unordered lists, because such lists are difficult to navigate with a screen reader. As a general content standard, DITA cannot prohibit such nesting. However, implementations can prevent such nesting through business processes or rule-based processing such as Schematron.

Another common accessibility recommendation is to avoid flashing or flickering video content. The DITA `<video>` element is a general mechanism for including video, and the content of that video is outside the scope of this specification.

Markup guidelines

Other accessibility guidelines require the use of specific DITA markup. Such guidelines are addressed in this specification.

For example, a requirement that images specify alternate text requires the use of the `<alt>` element within the `<image>` element. However, a guideline that the alternate text be *meaningful* is not something that can be enforced by DITA markup.

Guidelines that require enablement by DITA processors

Some accessibility guidelines require processors to take advantage of specific DITA markup.

Authors can use specific markup to enable accessible output. For example, by specifying a header row in a table, an author can define a header for every cell in the table body. However, to make the relationship between the table cell and header cell specific in a rendered format like HTML, the processor must make those relationships explicit in the output.

Processor requirements outside the scope of DITA markup

Processors have many other accessibility concerns that are outside the scope of this specification.

For example, WCAG has a requirement for contrast ratios when rendering substantive content. That requirement is unrelated to the source content. Such requirements apply to rendering mechanisms such as the CSS that is used to style DITA content in a browser.

As another example, a DITA processor might generate automated headings or include characters in output, such as:

- A section heading for an element specialized from `<section>`, such as "Requirements" for the `<prereq>` element in a task topic
- The greater-than character (>) that is typically used between phrases that are part of a menu navigation

It is up to the processor to use correct rendering for these cases, such as heading markup and accessible text alternatives for character displays such as the menu separator.

Related information

Web Content Accessibility Guidelines 2.0
The currently effective versions of WCAG are 2.0 and 2.1. WCAG 2.1 extends the 2.0 standard.

1.2 Accessible content

DITA provides elements and attributes that are designed to make content accessible.

Many common types of content are not accessible to all readers. For example, an image cannot be rendered by a screen reader, and a video cannot be rendered in many formats. DITA includes markup features that are designed to convey alternate versions of such content.

Alternate text for images

Alternate text is a textual description of an image. Systems often render the alternate text when the reader is using assistive technology or the image cannot be rendered.

The `<alt>` element is available inside of images as a way to specify alternate text.

Alternate text for areas of image maps

Within an image map, each defined area of the image can specify a cross reference. Whether or not the cross reference actually specifies a URI reference, the text within that cross reference functions as alternate or hover text.

Long descriptions for media

A long description reference is a reference to a textual description of a graphic or object. This is typically used to provide an extended description when the graphic or object is too complicated to describe with alternate text.

Processors can handle the reference in the following ways:

- Render the graphic or object as a link
- Make the extended description available to accessibility tools such as screen readers

While DITA provides the markup to enable these accessibility features, it is up to DITA processors to render output that uses the markup properly. For example, when a processor generates HTML5, alternate text must be specified using the `@alt` attribute on the `<img>` element.

1.3 Accessible tables

The complexity of table rendering requires authors and processors to be aware of several table-specific elements and attributes if they want to ensure that tables are accessible.

DITA topics support two types of tables: complex table and simple table.

The `<table>` element uses the OASIS Exchange Table Model, a simplification of the CALS table model. The complex table provides a wide variety of controls over the display properties of the data and even the table structure itself.

The <simpletable> element is structurally less complex than the <table> element and so is an easier base for specialization. It reflects a content model that is close to the HTML table. The <simpletable> element does not provide much control over formatting, although it permits titles and row and column spanning.

The following list provides information about table features that have an effect on table accessibility. Note that some features are applicable only to the complex tables that are produced by the <table> element.

Captions
Both table models allow for a caption to be provided by using the <title> element.

Cell headers
(Complex table only) When entries within a table function as headers, but do not fall into the categories of column or row headers, the @id and @headers attributes on <table> cells can be used. Specifying the @id attribute on the cell that functions as a header, and setting the @headers attribute to that ID value on the table cell for which it acts as a header serves to relate table cells to headers.

Column headers

Column headers are created using a header row, where each cell in the header row provides a header for other cells in its column. Both table models provide support for column headers:

Complex table
The <thead> element can provide one or more header rows.

Simple table
The <sthead> element can provide a single row header.

Row headers

Row headers are created using a header column, where each cell in the header column provides a header for other cells in its row. Both table models provide support for row headers:

Complex table
(First column) The @rowheader attribute can be set to "firstcol" on the <table> element to indicate that the first column is a header. Alternatively, the @scope attribute for each entry in the first column can be set to "row" to indicate that those cells are headers for their respective rows. (Other columns) The <colspec> element can define which columns function as headers. For that case, set the @rowheader attribute to "headers" on the column or columns that function as headers. Alternatively, set the @scope attribute on each relevant entry in the column to "row", indicating that the entry is a header for the entire row.

Simple table
The @keycol attribute can be set to the number of the column that functions as a header.

Summaries
(Complex table only) While the @summary attribute on tables is deprecated in HTML5, the <desc> element within a <table> can be used to store a summary. Since the content of the <desc> element is typically rendered as part of the content flow when used within <table>, processors might need special configuration to support this usage.

1.4 Examples of DITA markup for accessibility

This section contains examples of how DITA markup facilitates accessibility.

Comment by Kristen J Eberlein on 09 June 2022

A couple of comments:

- This section does not include examples of simple tables and acccessibility. Do we need an example here? Can we simply point to the examples in the `<simpletable>` topic?
- The example topics about `<table>` were relocated from the element-reference `<table>` topic. They were reviewed as part of that content, but might need some slight tweak now that they are part of the "Accessibility" topic.

1.4.1 Example: Alternate text for an image

In this scenario, an image of a ticketing workflow also provides alternate text that describes the image.

The following code sample references an image and provides alternate text:

```
<image href="workflow.png">
  <alt>A workflow diagram that shows a ticketing workflow.
       The workflow states are described in the text.</alt>
</image>
```

1.4.2 Example: Alternate text for an image map

Comment by robander
We have an example in the imagemap.dita element reference topic, which already includes alternate text. Not sure it's the best example for alt text, but also not sure whether it makes sense to come up with a new one?

Comment by Kristen J Eberlein on 09 June 2022

I think we need a realistic example here in this section which focuses on accessibility. The `<imagemap>` topic is in the "Utilities domain," which has not yet been reviewed. The example there is old and not very good; it dates back to DITA 1.0.

Here are some possible ideas for what to showcase:

- A simple pie chart of application users by industry sectors
- A collage of photos of a company's executive team
- A diagram of a sampler quilt, with various blocks linked to pages that explain the specific blocks

1.4.3 Example: Fallback information for multimedia content

In this scenario, fallback content is provided for systems that cannot display multimedia content.

The referenced video provides an image as fallback. If a system does not support video, it will display the image `video-not-available.png`, which specifies its own alternate text.

```
<video height="300px"
               loop="false"
               muted="false"
               poster="demo1-video-poster"
               width="400px">
  <desc>A video that illustrates how to conduct a system health scan.</desc>
  <fallback>
    <image href="video-not-available.png">
      <alt>This video cannot be displayed.</alt>
    </image>
  </fallback>
  <media-source href="video.mp4" format="video/mp4"/>
</video>
```

1.4.4 Example: Complex table with accessibility markup

In the following code sample, the table uses the `<thead>` element to identify header rows and the `@rowheader` attribute to identify a header column. These header relationships can be used to automatically create renderings of the table in other formats, such as HTML, that can be navigated using a screen reader or other assistive technology.

```
<table frame="all" rowheader="firstcol">
  <title>Sample of automated table accessibility</title>
  <desc>Names are listed in the column c1. Points are listed in both data columns, with
        expected points in column c2 and actual points in column c3.</desc>
  <tgroup cols="3">
    <colspec colname="c1"/>
    <colspec colname="c2"/>
    <colspec colname="c3"/>
    <thead>
      <row>
        <entry morerows="1">Name</entry>
        <entry namest="c2" nameend="c3">Points</entry>
      </row>
      <row>
        <entry>Expected</entry>
        <entry>Actual</entry>
      </row>
    </thead>
    <tbody>
      <row>
        <entry>Mark</entry>
        <entry>10,000</entry>
        <entry>11,123.45</entry>
      </row>
      <row>
        <entry>Peter</entry>
        <entry>9,000</entry>
        <entry>11,012.34</entry>
      </row>
      <row>
        <entry>Cindy</entry>
        <entry>10,000</entry>
        <entry>10,987.64</entry>
      </row>
     </tbody>
   </tgroup>
</table>
```

In this code sample, navigation information for assistive technology is derived from two sources:

- The `<thead>` element contains two rows, and indicates that each entry in those header rows is a header cell for that column. This means that each body cell can be associated with the header cell or cells above the column. For example, in the second body row, the entry "Peter" is associated with the header "Name"; similarly, the entry "9,000" is associated with the headers "Expected" and "Points".
- The `@rowheader` attribute that is specified on `<table>` indicates that the first column plays a role as a row header. This means that the header cell in column one is associated with the other body cells in the same row. For example, in the second body row, the entry "9,000" is associated with the header "Peter".

As a result of these two sets of headers, a rendering of the table associates the entry "9,000" with three headers: "Peter", "Expected", and "Points", thus making it fully navigable by a screen reader or other assistive technology. When the user navigates to the cell containing "9,000", it can report the headers "Peter", "Expected", and "Points" as the headers for that cell.

The output might be rendered in the following way:

Name	Points	
	Expected	Actual
Mark	10,000	11,123.45
Peter	9,000	11,012.34
Cindy	10,000	10,987.64

The rendered HTML used by a screen reader might look as follows.

```
<table>
  <caption>
    <span>Sample of automated table accessibility</span>
    <span class="desc">Names are listed in the column c1. Points are listed in both data
columns,
      with expected points in column c2 and actual points in column c3.</span>
  </caption>
  <colgroup><col><col><col></colgroup>
  <thead>
    <tr>
      <th id="source__entry__1" rowspan="2">Name</th>
      <th id="source__entry__2" colspan="2">Points</th>
    </tr>
    <tr>
      <th id="source__entry__3">Expected</th>
      <th id="source__entry__4">Actual</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <th scope="row" id="source__entry__5" headers="source__entry__1">Mark</th>
      <td headers="source__entry__5 source__entry__2 source__entry__3">10,000</td>
      <td headers="source__entry__5 source__entry__2 source__entry__4">11,123.45</td>
    </tr>
    <tr>
      <th scope="row" id="source__entry__8" headers="source__entry__1">Peter</th>
      <td headers="source__entry__8 source__entry__2 source__entry__3">9,000</td>
      <td headers="source__entry__8 source__entry__2 source__entry__4">11,012.34</td>
    </tr>
    <tr>
      <th scope="row" id="source__entry__11" headers="source__entry__1">Cindy</th>
      <td headers="source__entry__11 source__entry__2 source__entry__3">10,000</td>
      <td headers="source__entry__11 source__entry__2 source__entry__4">10,987.64</td>
    </tr>
  </tbody>
</table>
```

1.4.5 Example: Complex table with some manually-specified accessibility markup

In some complex tables, the `<thead>` element and `@rowheader` attribute might not be enough to support all accessibility needs. Assume that a table is designed so that names are listed across the top row, instead of in the first column, with both the first and second columns also functioning as headers:

Name		Mark	Peter	Cindy
Points	Expected	10,000	9,000	10,000
	Actual	11,123.45	11,012.34	10,987.64

Note The table in this example is not meant to illustrate a best practice; this specific example would likely prove difficult to navigate using a screen reader even with proper header markup. This example is only intended to illustrate the full range of manual accessibility markup that is available should the need arise.

Here, the `@rowheader` attribute cannot be used, because it is only able to specify the first column as a header column. In this case, the `@scope` attribute can be used to indicate that entries in the first and second columns function as headers for the entire row (or row group, in the case of a cell that spans more than one row).

The following code sample demonstrates the use of the `@scope` attribute to facilitate navigation of these rows by a screen reader or other assistive technology. Note that the `<thead>` element is still used to imply a header relationship with the names at the top of each column.

```
<table frame="all">
 <title>Sample with two header columns</title>
 <tgroup cols="5">
  <colspec colname="c1"/>
  <colspec colname="c2"/>
  <colspec colname="c3"/>
  <colspec colname="c4"/>
  <colspec colname="c5"/>
  <thead>
   <row>
    <entry namest="c1" nameend="c2">Name</entry>
    <entry>Mark</entry>
    <entry>Peter</entry>
    <entry>Cindy</entry>
   </row>
  </thead>
  <tbody>
   <row>
    <entry morerows="1" scope="rowgroup"><b>Points</b></entry>
    <entry scope="row"><b>Expected</b></entry>
    <entry>10,000</entry>
    <entry>9,000</entry>
    <entry>10,000</entry>
   </row>
   <row>
    <entry scope="row"><b>Actual</b></entry>
    <entry>11,123.45</entry>
    <entry>11,012.34</entry>
    <entry>10,987.64</entry>
   </row>
  </tbody>
 </tgroup>
</table>
```

The rendered HTML used by a screen reader might look as follows.

```
<table>
  <caption>Sample with two header columns</caption>
  <colgroup><col><col><col><col><col></colgroup>
  <thead>
    <tr>
      <th id="source__entry__1" colspan="2">Name</th>
      <th id="source__entry__2">Mark</th>
      <th id="source__entry__3">Peter</th>
      <th id="source__entry__4">Cindy</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <th headers="source__entry__1" rowspan="2" scope="rowgroup"><strong class="ph
b">Points</strong></th>
      <th headers="source__entry__1" scope="row"><strong class="ph b">Expected</strong></th>
      <td headers="source__entry__2">10,000</td>
      <td headers="source__entry__3">9,000</td>
```

```
      <td headers="source__entry__4">10,000</td>
    </tr>
    <tr>
      <th headers="source__entry__1" scope="row"><strong class="ph b">Actual</strong></th>
      <td headers="source__entry__2">11,123.45</td>
      <td headers="source__entry__3">11,012.34</td>
      <td headers="source__entry__4">10,987.64</td>
    </tr>
  </tbody>
</table>
```

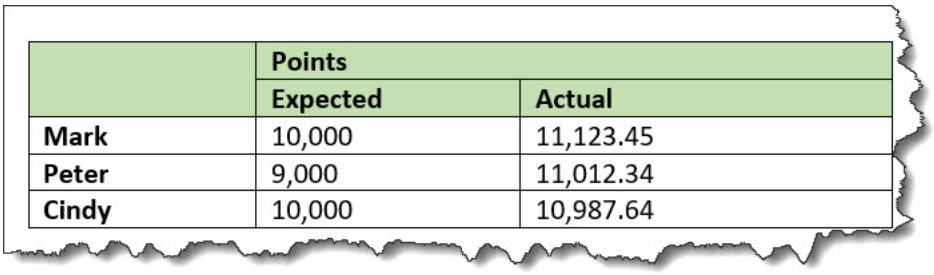
1.4.6 Example: Complex table with manual accessibility markup

In extremely complex tables, such as those with a single header cell in the middle of the table, fine-grained accessibility controls are available to explicitly associate any content cell with any header cell. This might also be useful for cases where processors do not support implied accessibility relationships that exist based on header markup such as `<thead>`.

In the following sample, header cells are identified using the `@id` attribute, which is referenced using the `@headers` attribute on appropriate content cells. This makes all header relationships in the table explicit. Note that this sample ignores the `@scope` attribute, which could be used to exercise manual control without setting as many attribute values; it also ignores the fact that `<thead>` creates a header relationship even when the `@id` and `@headers` attributes are not used.

```
<table frame="all">
  <title>Sample with fully manual accessibility control</title>
<desc>Names are listed in the column c1. Points are listed in both data columns, with
expected points in column c2 and actual points in column c3.</desc>
  <tgroup cols="3">
    <colspec colname="c1"/>
    <colspec colname="c2"/>
    <colspec colname="c3"/>
    <thead>
      <row>
        <entry morerows="1"> </entry>
        <entry namest="c2" nameend="c3" id="pts">Points</entry>
      </row>
      <row>
        <entry id="exp" headers="pts">Expected</entry>
        <entry id="act" headers="pts">Actual</entry>
      </row>
    </thead>
    <tbody>
      <row>
        <entry id="name1">Mark</entry>
        <entry headers="name1 exp pts">10,000</entry>
        <entry headers="name1 act pts">11,123.45</entry>
      </row>
      <row>
        <entry id="name2">Peter</entry>
        <entry headers="name2 exp pts">9,000</entry>
        <entry headers="name2 act pts">11,012.34</entry>
      </row>
      <row>
        <entry id="name3">Cindy</entry>
        <entry headers="name3 exp pts">10,000</entry>
        <entry headers="name3 act pts">10,987.64</entry>
      </row>
     </tbody>
   </tgroup>
</table>
```

The output might be rendered in the following way:



	Points	
	Expected	Actual
Mark	10,000	11,123.45
Peter	9,000	11,012.34
Cindy	10,000	10,987.64

The rendered HTML used by a screen reader might look as follows.

```
<table>
  <caption>Sample with fully manual accessibility control
           <span class="desc">Names are listed in the column c1. Points are listed in both
data columns, with
                expected points in column c2 and actual points in column c3.</span></caption>
  <colgroup><col><col><col></colgroup>
  <thead>
    <tr>
      <th id="entry__1" rowspan="2"> </th>
      <th id="pts" colspan="2">Points</th>
    </tr>
    <tr>
      <th id="exp" headers="pts">Expected</th>
      <th id="act" headers="pts">Actual</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td id="name1" headers="entry__1">Mark</td>
      <td headers="name1 pts exp">10,000</td>
      <td headers="name1 pts act">11,123.45</td>
    </tr>
    <tr>
      <td id="name2" headers="entry__1">Peter</td>
      <td headers="name2 pts exp">9,000</td>
      <td headers="name2 pts act">11,012.34</td>
    </tr>
    <tr>
      <td id="name3" headers="entry__1">Cindy</td>
      <td headers="name3 pts exp">10,000</td>
      <td headers="name3 pts act">10,987.64</td>
    </tr>
  </tbody>
</table>
```

A Aggregated RFC-2119 statements

This appendix contains all the normative statements from the DITA 2.0 specification. They are aggregated here for convenience in this non-normative appendix.

Index

A