Tagged PDF, PDF/A and PDF/UA Compliance

1. Tagged PDF

Tagged PDF files contain information about the structure of the document. The information about the structure is transported via so-called "PDF tags". Tagging a PDF makes it more accessible to screen readers and other accessibility tools. It contains important information like the languages of texts, the structures of tables and alternative texts for graphics.

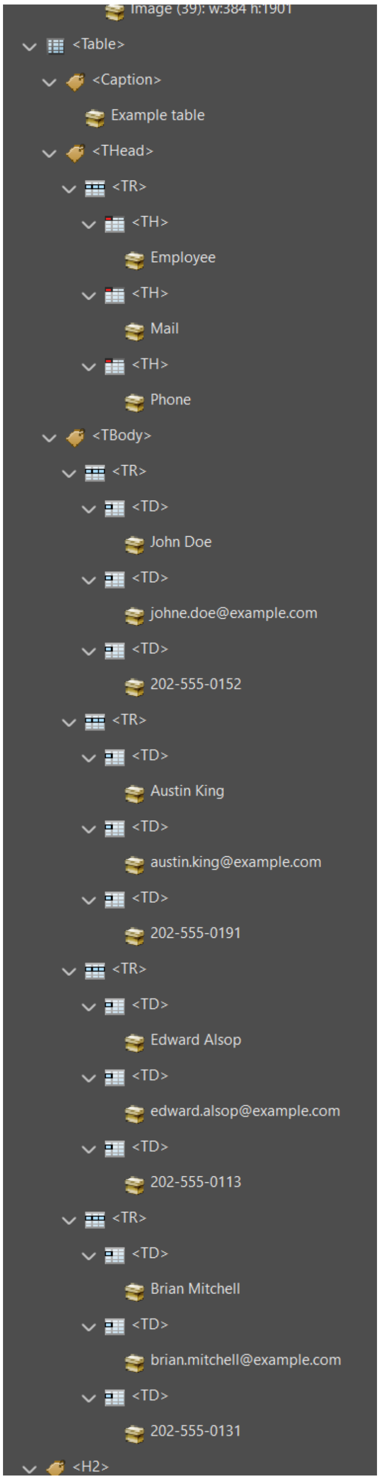
Enabled tagging can also improve copy and paste behavior. For example, copying paragraphs from a tagged PDF created with PDFreactor can ignore the implicit line breaks inside the paragraphs. This information also allows for reflow in viewers that support it.

Using the `addTags` configuration property, you can add PDF tags to PDF documents generated with PDFreactor. If you are generating PDFs from HTML documents, the HTML elements and their styles are automatically mapped to the corresponding PDF tags, so all you have to do is set this property to enable tagging.

The screenshot on the right (taken from Adobe Acrobat DC) shows that PDFreactor is capable of tagging even complex structures such as tables properly. The table below was placed at the bottom of the page to demonstrate that PDFreactor won't repeat the <table> or <thead> tag even though the table splits onto another page, repeating its header. The continuation markers are also ignored as pagination artifacts, as are page headers and footers.

The fully automated tagging can be overridden using WAI-ARIA (see related chapter) or various custom CSS Properties `-ro-pdf-tag-*` (see PDFreactor manual), allowing for as much, or as little, manual intervention as required.

A tagged PDF will often be bigger than an equivalent PDF file that does not include PDF tags, especially when full compression is disabled. Please note that PDF/A-1 conformance disables full compression.



```
        Image (39): w:384 h:1901
  <Table>
    <Caption>
      Example table
    <THead>
      <TR>
        <TH>
          Employee
        <TH>
          Mail
        <TH>
          Phone
    <TBody>
      <TR>
        <TD>
          John Doe
        <TD>
          johne.doe@example.com
        <TD>
          202-555-0152
      <TR>
        <TD>
          Austin King
        <TD>
          austin.king@example.com
        <TD>
          202-555-0191
      <TR>
        <TD>
          Edward Alsop
        <TD>
          edward.alsop@example.com
        <TD>
          202-555-0113
      <TR>
        <TD>
          Brian Mitchell
        <TD>
          brian.mitchell@example.com
        <TD>
          202-555-0131
  <H2>
```

Example table

Employee	Mail	Phone
John Doe	johne.doe@example.com	202-555-0152
Austin King	austin.king@example.com	202-555-0191

Continued on next page

Continued from previous page

Employee	Mail	Phone
Edward Alsop	edward.alsop@example.com	202-555-0113
Brian Mitchell	brian.mitchell@example.com	202-555-0131

1. PDF/A Conformance

PDF/A differs from PDF by prohibiting features ill-suited to long-term archiving, such as font linking (as opposed to font embedding).

The PDF/A standard does not define an archiving strategy or the goals of an archiving system. It identifies a "profile" for electronic documents that ensures the documents can be reproduced exactly the same way using various software in years to come. A key element to this reproducibility is the requirement for PDF/A documents to be 100% self-contained. All of the information necessary for displaying the document in the same manner is embedded in the file. This includes, but is not limited to, all content (text, raster images and vector graphics), fonts and color information. A PDF/A document is not permitted to be reliant on information from external sources (e.g. font programs and data streams), but may include annotations (e.g. hypertext links) that link to external documents.

PDFreactor supports the creation of all PDF/A conformant files.

Many companies and government organizations worldwide require PDF/A conformant documents. Tagged PDFs are a requirement of Section 508 of the American Rehabilitation Act.

PDF/A-1a is the most strict PDF/A standard while the newer PDF/A standards are more lenient, e.g. allowing transparency and attachments.

1.1. Common PDF/A conformance requirements

PDF/A restriction	PDFreactor actions
All used fonts are embedded.	PDFreactor ignores the option to disable font embedding.
All images are embedded.	Images are always automatically embedded by PDFreactor.
Multi-media content is prohibited.	Embedding objects is automatically prevented by PDFreactor when PDF/A conformance is set.
JavaScript is prohibited.	No JavaScript is embedded when PDF/A conformance is set. (This does not prohibit JavaScript in the source HTML document to be processed during conversions)
Encryption is disallowed.	This is automatically prevented when the PDF/A conformance is set.
The PDF must be tagged.	This is automatically done by PDFreactor when PDF/A conformance is set.
Metadata included in the PDF is required to be standard-based XMP.	This is automatically done by PDFreactor when PDF/A conformance is set.

PDF/A restriction	PDFreactor actions
Colors are specified in a device-independent manner.	In PDFreactor colors are defined either as RGB or CMYK. When PDF/A conformance is set, one of these color spaces has to be set in conjunction with a color space profile. CMYK requires an ICC profile to be set, RGB colors use a default sRGB profile, if no other is set. Using RGB colors in CMYK PDF/A documents or vice versa is prohibited. Color keywords and shades specified via the "gray" function are converted to the appropriate color space losslessly.
Requires PDF version 1.4, specifically.	PDFreactor automatically disables features that require PDF 1.5 or newer, incl. full compression, which will increase the size of the output files, especially as they are tagged.

1.2. PDF/A-1a specific conformance requirements

PDF/A-1a restriction	PDFreactor actions
Transparency is disallowed.	PDFreactor will ignore certain kinds of transparency of images. Other occurrences of transparency will cause an exception to be thrown.
Attachments are disallowed.	This is automatically prevented when PDF/A-1a conformance is set.

To create a PDF/A conformant document, the configuration property `conformance` can be used in the PDFreactor integration:

```
config.setConformance(Conformance.PDFA3A);
```

If CMYK colors are used in a document to be converted into a PDF/A-conformant file, an Output Intent has to be set. This is possible to use the following API calls:

```
Configuration config = new Configuration();
OutputIntent outputIntent = new OutputIntent();
outputIntent.setIdentifier("ICC profile identifier");

// Use this if you are loading the ICC profile via URL
outputIntent.setUrl("URL/to/ICC/profile");

// Use this if you want to specify the ICC profile's binary data
outputIntent.setData(iccProfileBinaryData);
config.setOutputIntent(outputIntent);
```

The `identifier` property is a string identifying the intended output device or production condition in human- or machine-readable form. The `url` property points to an ICC profile file while the `data` property contains data of such a profile.

Note

When PDF/A conformance is set, encryption, restrictions, comments, full compression and other non PDF/A-conformant features are automatically overridden, regardless of their own settings.

Setting PDF/A-1a conformance generates PDFs with Adobe PDF version 1.4 in which some PDF tags are forbidden e.g. <tbody>. PDFreactor will skip all forbidden tags automatically, but handle table headers correctly.

2. PDF/UA Conformance

PDF/UA (PDF/Universal Accessibility) is the informal name for ISO 14289, the International Standard for accessible PDF technology. A technical specification intended for developers implementing PDF writing and processing software, PDF/UA provides definitive terms and requirements for accessibility in PDF documents and applications. For those equipped with appropriate software, conformance with PDF/UA ensures accessibility for people with disabilities who use assistive technology such as screen readers, screen magnifiers, joysticks and other technologies to navigate and read electronic content.

PDF/UA can be combined with PDF/A to create PDFs that are conformant with both standards simultaneously. For this, PDFreactor offers combined conformance constants like this:

```
config.setConformance(Conformance.PDFA3A_PDFUA1);
```

3. Tagging PDFs using WAI-ARIA attributes

PDFreactor 12 adds the ability to translate ARIA attributes into appropriate PDF tags. As opposed to the prior parts of this sample, the following pages take advantage of this.

There will be different cases that alternate between their content as it is rendered in the PDF, their respective source code as well as resulting accessibility tree when viewed in the Google Chrome Devtools in comparison to the resulting PDF viewed in PAC. Note however, that the embedded pictures do not contain all embedded accessibility information.

To see all tagged aspects and attributes, you can inspect this PDF document with a tool that can analyze accessible PDFs like the PDF accessibility checker (PAC) by the PDF/UA Foundation.

For more detailed information about the usage of ARIA in PDFreactor, like the exact mapping of roles to PDF tags, as well as the creation of accessible documents in general, please refer to the respective chapter in the manual.

Note

The HTML elements in these examples don't always match their ARIA roles. This is by design, as PDFreactor would already tag most of them appropriately otherwise, without the addition of ARIA attributes.
The source and accessibility data of these examples is not tagged to make the tag tree more easy to navigate.

Important

Even though WAI-ARIA attributes can be mapped to PDF tag structures automatically, the resulting PDFs should be validated independently from the source HTML document.

Example 1: Tagging a table

The following section is tagged as a table, including column headers. The table has a size of 2 by 5 cells.

View

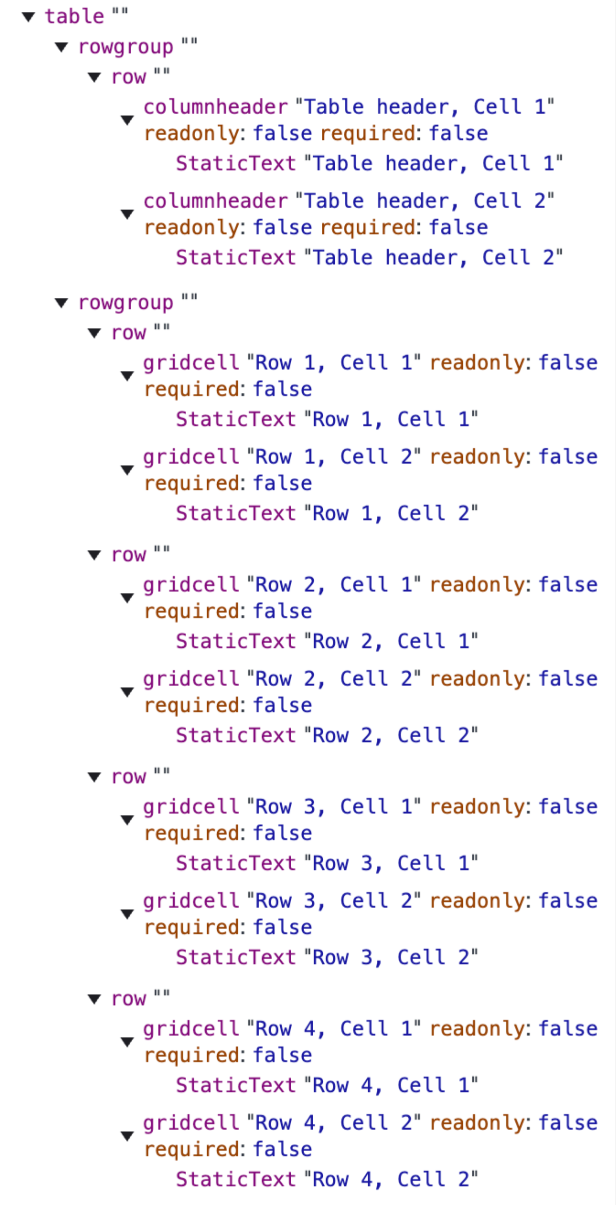
Table header, Cell 1 Table header, Cell 2
Row 1, Cell 1 Row 1, Cell 2
Row 2, Cell 1 Row 2, Cell 2
Row 3, Cell 1 Row 3, Cell 2
Row 4, Cell 1 Row 4, Cell 2

Source

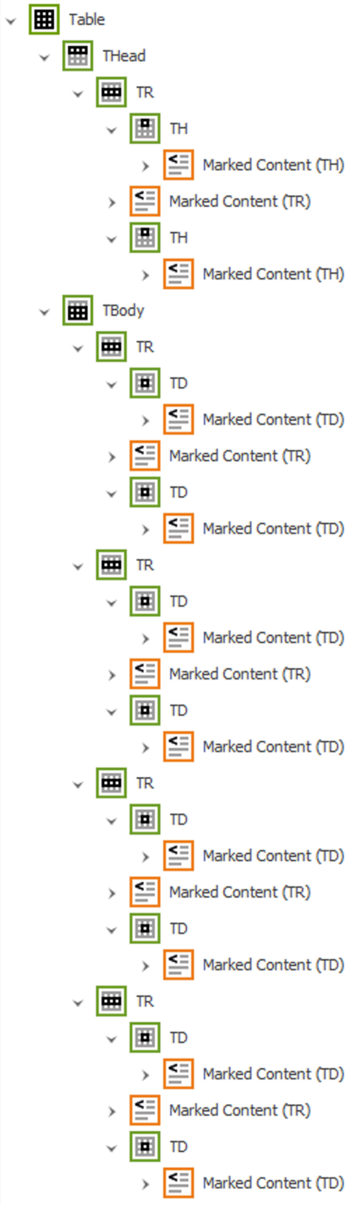
```
<div role="table">
    <div role="rowgroup">
        <div role="row">
            <span role="columnheader">Table header, Cell 1 </span><span role="columnheader">
                    Table header, Cell 2 </span>
        </div>
    </div>
    <div role="rowgroup">
        <div role="row">
            <span role="cell">Row 1, Cell 1 </span><span role="cell">Row 1, Cell 2 </span>
        </div>
        <div role="row">
            <span role="cell">Row 2, Cell 1 </span><span role="cell">Row 2, Cell 2 </span>
        </div>
        <div role="row">
            <span role="cell">Row 3, Cell 1 </span><span role="cell">Row 3, Cell 2 </span>
        </div>
        <div role="row">
            <span role="cell">Row 4, Cell 1 </span><span role="cell">Row 4, Cell 2 </span>
        </div>
    </div>
</div>
```

Example 1: Tagging a table

Google Chrome | **Accessibility Information** | **PDF**



```
▼ table ""
  ▼ rowgroup ""
    ▼ row ""
      ▼ columnheader "Table header, Cell 1"
        readonly: false required: false
          StaticText "Table header, Cell 1"
      ▼ columnheader "Table header, Cell 2"
        readonly: false required: false
          StaticText "Table header, Cell 2"
  ▼ rowgroup ""
    ▼ row ""
      ▼ gridcell "Row 1, Cell 1" readonly: false
        required: false
          StaticText "Row 1, Cell 1"
      ▼ gridcell "Row 1, Cell 2" readonly: false
        required: false
          StaticText "Row 1, Cell 2"
    ▼ row ""
      ▼ gridcell "Row 2, Cell 1" readonly: false
        required: false
          StaticText "Row 2, Cell 1"
      ▼ gridcell "Row 2, Cell 2" readonly: false
        required: false
          StaticText "Row 2, Cell 2"
    ▼ row ""
      ▼ gridcell "Row 3, Cell 1" readonly: false
        required: false
          StaticText "Row 3, Cell 1"
      ▼ gridcell "Row 3, Cell 2" readonly: false
        required: false
          StaticText "Row 3, Cell 2"
    ▼ row ""
      ▼ gridcell "Row 4, Cell 1" readonly: false
        required: false
          StaticText "Row 4, Cell 1"
      ▼ gridcell "Row 4, Cell 2" readonly: false
        required: false
          StaticText "Row 4, Cell 2"
```



```
⌄ Table
  ⌄ THead
    ⌄ TR
      ⌄ TH
        › Marked Content (TH)
      › Marked Content (TR)
      ⌄ TH
        › Marked Content (TH)
  ⌄ TBody
    ⌄ TR
      ⌄ TD
        › Marked Content (TD)
      › Marked Content (TR)
      ⌄ TD
        › Marked Content (TD)
    ⌄ TR
      ⌄ TD
        › Marked Content (TD)
      › Marked Content (TR)
      ⌄ TD
        › Marked Content (TD)
    ⌄ TR
      ⌄ TD
        › Marked Content (TD)
      › Marked Content (TR)
      ⌄ TD
        › Marked Content (TD)
    ⌄ TR
      ⌄ TD
        › Marked Content (TD)
      › Marked Content (TR)
      ⌄ TD
        › Marked Content (TD)
```

Example 2: Tagging a table of contents

This section is tagged as a table of contents or list with 16 entries and up to 4 levels.

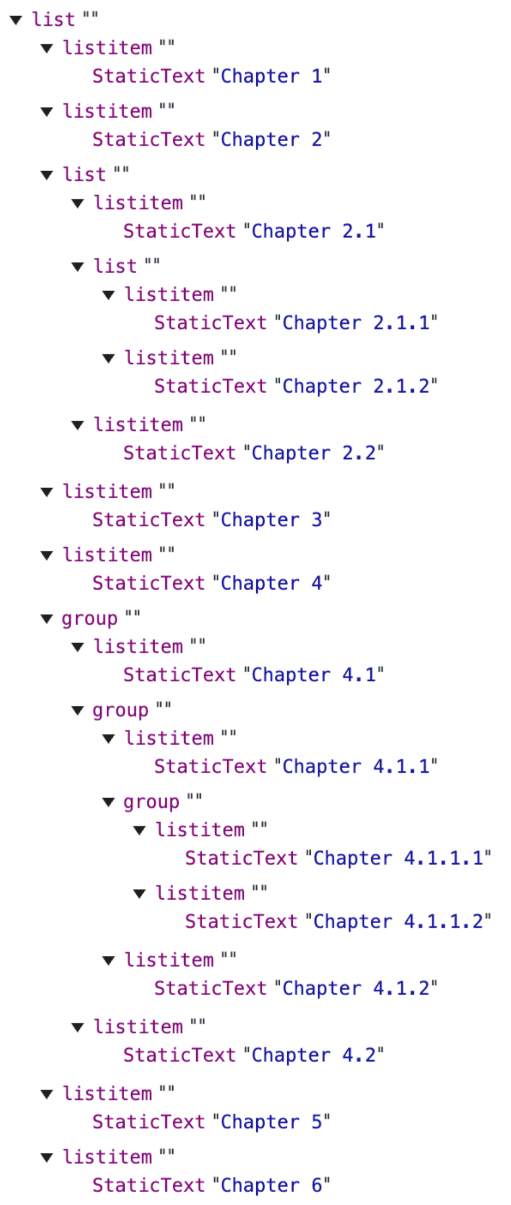
View

Chapter 1
Chapter 2
Chapter 2.1
Chapter 2.1.1
Chapter 2.1.2
Chapter 2.2
Chapter 3
Chapter 4
Chapter 4.1
Chapter 4.1.1
Chapter 4.1.1.1
Chapter 4.1.1.2
Chapter 4.1.2
Chapter 4.2
Chapter 5
Chapter 6

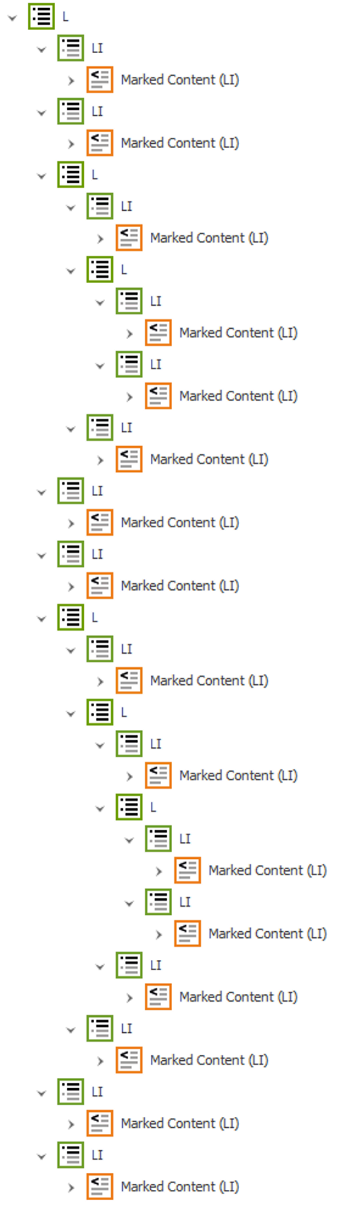
Source

```
<div role="list">
    <div role="listitem">Chapter 1</div>
    <div role="listitem">Chapter 2</div>
    <div role="list">
        <div role="listitem">Chapter 2.1</div>
        <div role="list">
            <div role="listitem">Chapter 2.1.1</div>
            <div role="listitem">Chapter 2.1.2</div>
        </div>
        <div role="listitem">Chapter 2.2</div>
    </div>
    <div role="listitem">Chapter 3</div>
    <div role="listitem">Chapter 4</div>
    <div role="group">
        <div role="listitem">Chapter 4.1</div>
        <div role="group">
            <div role="listitem">Chapter 4.1.1</div>
            <div role="group">
                <div role="listitem">Chapter 4.1.1.1</div>
                <div role="listitem">Chapter 4.1.1.2</div>
            </div>
            <div role="listitem">Chapter 4.1.2</div>
        </div>
        <div role="listitem">Chapter 4.2</div>
    </div>
    <div role="listitem">Chapter 5</div>
    <div role="listitem">Chapter 6</div>
</div>
```

Google Chrome | Accessibility Information | PDF



```
▼ list ""
   ▼ listitem ""
        StaticText "Chapter 1"
   ▼ listitem ""
        StaticText "Chapter 2"
   ▼ list ""
      ▼ listitem ""
           StaticText "Chapter 2.1"
      ▼ list ""
         ▼ listitem ""
              StaticText "Chapter 2.1.1"
         ▼ listitem ""
              StaticText "Chapter 2.1.2"
      ▼ listitem ""
           StaticText "Chapter 2.2"
   ▼ listitem ""
        StaticText "Chapter 3"
   ▼ listitem ""
        StaticText "Chapter 4"
   ▼ group ""
      ▼ listitem ""
           StaticText "Chapter 4.1"
      ▼ group ""
         ▼ listitem ""
              StaticText "Chapter 4.1.1"
         ▼ group ""
            ▼ listitem ""
                 StaticText "Chapter 4.1.1.1"
            ▼ listitem ""
                 StaticText "Chapter 4.1.1.2"
         ▼ listitem ""
              StaticText "Chapter 4.1.2"
      ▼ listitem ""
           StaticText "Chapter 4.2"
   ▼ listitem ""
        StaticText "Chapter 5"
   ▼ listitem ""
        StaticText "Chapter 6"
```

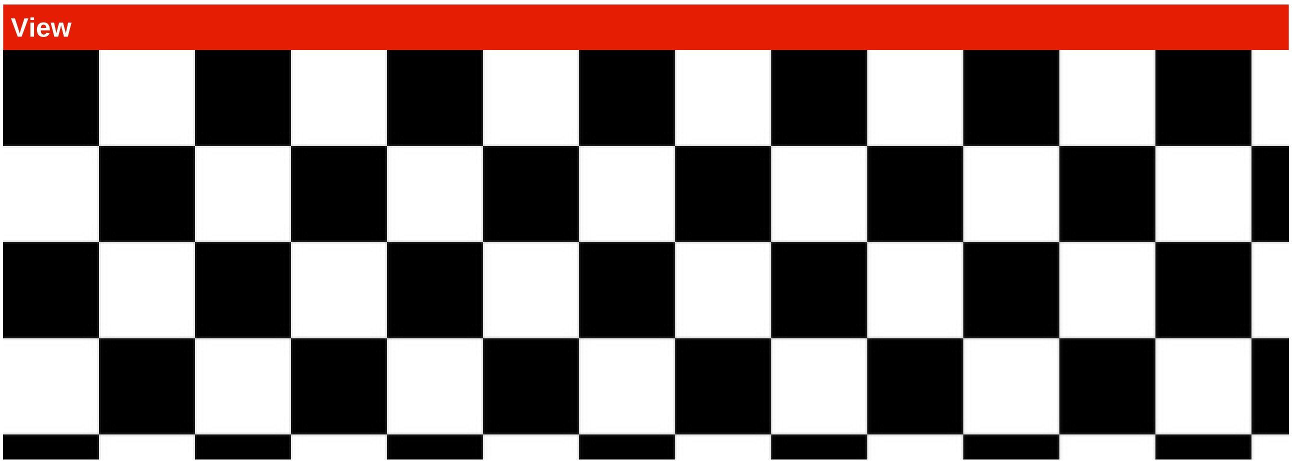


```
L
  LI
    Marked Content (LI)
  LI
    Marked Content (LI)
  L
    LI
      Marked Content (LI)
    L
      LI
        Marked Content (LI)
      LI
        Marked Content (LI)
    LI
      Marked Content (LI)
  LI
    Marked Content (LI)
  LI
    Marked Content (LI)
  L
    LI
      Marked Content (LI)
    L
      LI
        Marked Content (LI)
      L
        LI
          Marked Content (LI)
        LI
          Marked Content (LI)
      LI
        Marked Content (LI)
    LI
      Marked Content (LI)
  LI
    Marked Content (LI)
  LI
    Marked Content (LI)
```

Example 3: Labeling an image

The following element is tagged as an image.

View



Source

```
<div aria-label="Checkerboard pattern" role="img"
    style="background-image: url('./img/checkerboard.jpg'); height: 5cm;"></div>
```

This element is represented as 'img' in Chrome and tagged as a 'Figure' in PDF.

Example 4: Tagging forms

This section is tagged as if it contained forms: A textbox, a "mixed" checkbox and a radio group with 3 buttons.

Note

Only the tags for non-interactive forms can be set like this. In case interactive PDF forms/AcroForms are used, their `role/type` and `checked` values can't be overridden this way.

View

Textbox content

/

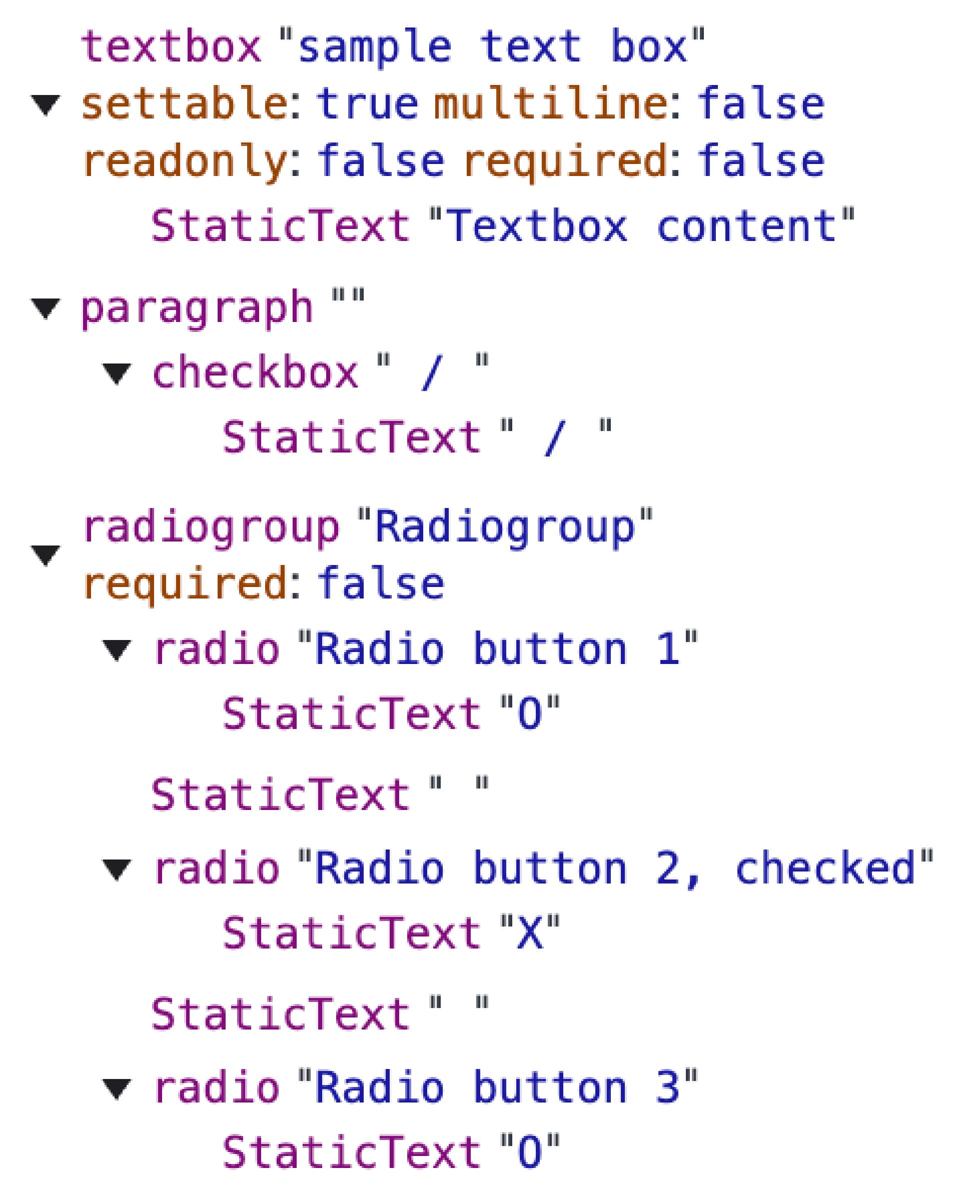
O X O

Source

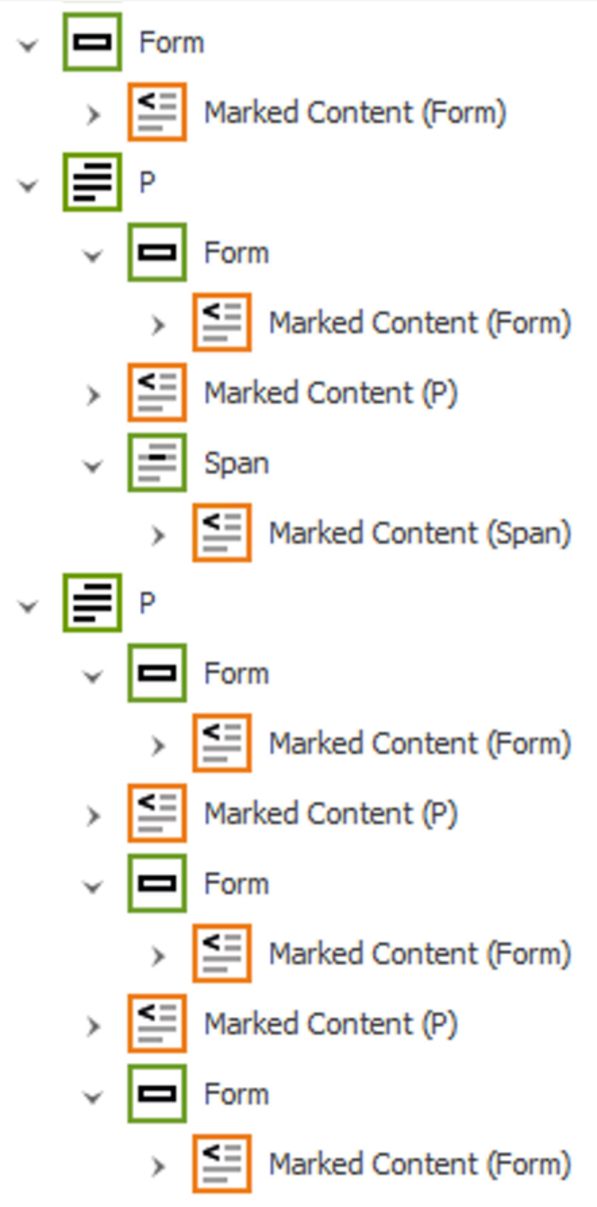
```
<p aria-label="sample text box" role="textbox">Textbox content</p>
<p>
    <span aria-checked="mixed" role="checkbox"> / </span>
</p>
<p aria-label="Radiogroup" role="radiogroup">
    <span aria-label="Radio button 1" role="radio">O</span>
    <span aria-checked="true" aria-label="Radio button 2, checked" role="radio">X</span>
    <span aria-label="Radio button 3" role="radio">O</span>
</p>
```

Example 4: Tagging forms

Google Chrome | Accessibility Information | PDF



```
  textbox "sample text box"
▼ settable: true multiline: false
  readonly: false required: false
      StaticText "Textbox content"

▼ paragraph ""
    ▼ checkbox " / "
          StaticText " / "

  radiogroup "Radiogroup"
▼ required: false
    ▼ radio "Radio button 1"
          StaticText "O"

      StaticText " "

    ▼ radio "Radio button 2, checked"
          StaticText "X"

      StaticText " "

    ▼ radio "Radio button 3"
          StaticText "O"
```



```
⌄ Form
    › Marked Content (Form)
⌄ P
    ⌄ Form
        › Marked Content (Form)
    › Marked Content (P)
    ⌄ Span
        › Marked Content (Span)
⌄ P
    ⌄ Form
        › Marked Content (Form)
    › Marked Content (P)
    ⌄ Form
        › Marked Content (Form)
    › Marked Content (P)
    ⌄ Form
        › Marked Content (Form)
```

Example 5: Miscellaneous ARIA attributes

This section showcases some of the ARIA attributes that can be used to convey additional information to assistive technology. You can access labels and other data that is not directly visible in the accessibility tree by viewing this HTML document in a browser or by checking the PDF using a suitable tool.

View

Although this paragraph is clearly visible, it is marked as an artifact due to its `aria-hidden` attribute being set to true.

The `aria-label` attribute allows you to attach a label to an user interface element using the attribute value, just as it was done for this checkbox: □

In the same way, `aria-labelledby` can be used to label them using the content of other elements: □

A label created from the content of another element.

`aria-describedby` is similar to `aria-labelledby`, with the key difference that labels are meant to be short and concise, while descriptions may provide additional information that might not be needed: □ However, both labels and descriptions are mapped to the same "Desc" attribute when mapped to PDF. When both `aria-describedby` and `aria-labelledby` are applied to the same element, the label takes precedence.

A checkbox without any purpose aside from being described by the content of another element. Note the additional length and verbosity because it is not just labelled but described.

Both `aria-describedby` and `aria-labelledby` need to refer to an element that is part of the DOM, meaning that it needs to be at least theoretically visible. Note that `display: none;` will remove the element.

This heading is tagged with level 4,

while this one with level 5. Both because of `aria-level`.

Note that these headings are automatically added to the bookmarks in the documents outline, just as any `h` element would be.

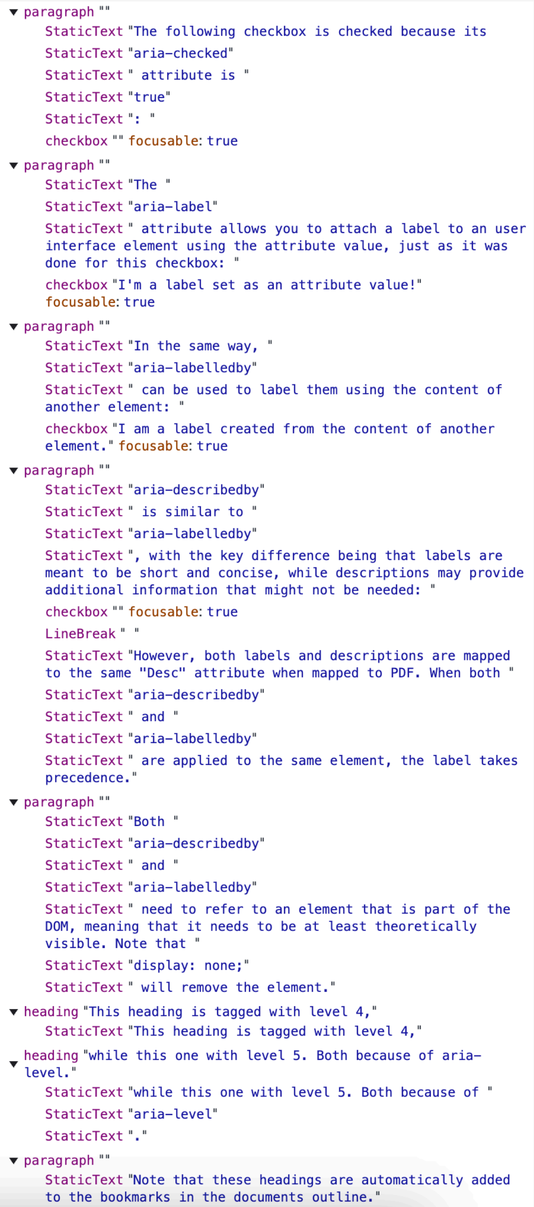
Example 5: Miscellaneous ARIA attributes

Source

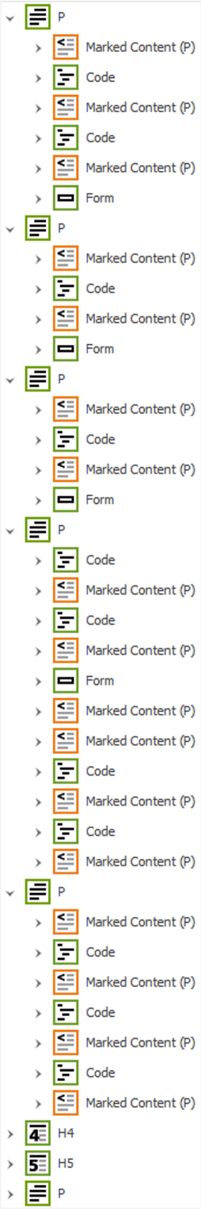
```
<p aria-hidden="true">Although this paragraph is clearly visible, it is marked as an artifact
    due to its <code>aria-hidden</code> attribute being set to true.
</p>

<p>The <code>aria-label</code> attribute allows you to attach a label to an user interface element
    using the attribute value, just as it was done for this checkbox:
    <input aria-label="A label set as an attribute value!" type="checkbox">
</p>

<p>In the same way, <code>aria-labelledby</code> can be used to label them using the content
    of another element: <input aria-labelledby="label-identifier" type="checkbox">
</p>

<p aria-hidden="true" id="label-identifier">A label created from the content
    of another element.</p>

<p><code>aria-describedby</code> is similar to <code>aria-labelledby</code>, with the key
    difference being that labels are meant to be short and concise, while descriptions
    may provide additional information that might not be needed:
    <input aria-describedby="description-identifier" type="checkbox">
    <br>
    However, both labels and descriptions are mapped to the same "Desc" attribute when mapped to
    PDF. When both <code>aria-describedby</code> and <code>aria-labelledby</code> are applied to
    the same element, the label takes precedence.
</p>

<p aria-hidden="true" id="description-identifier">A checkbox
    without any purpose aside from being described by the content of another element.
    Note the additional length and verbosity because it is not just labelled but described.</p>

<p>Both <code>aria-describedby</code> and <code>aria-labelledby</code> need to refer to an element
    that is part of the DOM, meaning that it needs to be at least theoretically visible.
    Note that <code>display: none;</code> will remove the element.</p>

<p aria-level="4" role="heading">This heading is tagged with level 4,</p>

<p aria-level="5" role="heading">while this one with level 5. Both because of
    <code>aria-level</code>.
</p>
<p>Note that these headings are automatically added to the bookmarks in the documents outline,
    just as any <code>h</code> element would be.</p>
```

Example 5: Miscellaneous ARIA attributes

Google Chrome — Accessibility Information — PDF



```
▼ paragraph ""
    StaticText "The following checkbox is checked because its
    StaticText "aria-checked"
    StaticText " attribute is "
    StaticText "true"
    StaticText ": "
    checkbox "" focusable: true
▼ paragraph ""
    StaticText "The "
    StaticText "aria-label"
    StaticText " attribute allows you to attach a label to an user
    interface element using the attribute value, just as it was
    done for this checkbox: "
    checkbox "I'm a label set as an attribute value!"
    focusable: true
▼ paragraph ""
    StaticText "In the same way, "
    StaticText "aria-labelledby"
    StaticText " can be used to label them using the content of
    another element: "
    checkbox "I am a label created from the content of another
    element." focusable: true
▼ paragraph ""
    StaticText "aria-describedby"
    StaticText " is similar to "
    StaticText "aria-labelledby"
    StaticText ", with the key difference being that labels are
    meant to be short and concise, while descriptions may provide
    additional information that might not be needed: "
    checkbox "" focusable: true
    LineBreak " "
    StaticText "However, both labels and descriptions are mapped
    to the same "Desc" attribute when mapped to PDF. When both "
    StaticText "aria-describedby"
    StaticText " and "
    StaticText "aria-labelledby"
    StaticText " are applied to the same element, the label takes
    precedence."
▼ paragraph ""
    StaticText "Both "
    StaticText "aria-describedby"
    StaticText " and "
    StaticText "aria-labelledby"
    StaticText " need to refer to an element that is part of the
    DOM, meaning that it needs to be at least theoretically
    visible. Note that "
    StaticText "display: none;"
    StaticText " will remove the element."
▼ heading "This heading is tagged with level 4,"
    StaticText "This heading is tagged with level 4,"
▼ heading "while this one with level 5. Both because of aria-
  level."
    StaticText "while this one with level 5. Both because of "
    StaticText "aria-level"
    StaticText "."
▼ paragraph ""
    StaticText "Note that these headings are automatically added
    to the bookmarks in the documents outline."
```



```
⌄ P
    › Marked Content (P)
    › Code
    › Marked Content (P)
    › Code
    › Marked Content (P)
    › Form
⌄ P
    › Marked Content (P)
    › Code
    › Marked Content (P)
    › Form
⌄ P
    › Marked Content (P)
    › Code
    › Marked Content (P)
    › Form
⌄ P
    › Code
    › Marked Content (P)
    › Code
    › Marked Content (P)
    › Form
    › Marked Content (P)
    › Marked Content (P)
    › Code
    › Marked Content (P)
    › Code
    › Marked Content (P)
⌄ P
    › Marked Content (P)
    › Code
    › Marked Content (P)
    › Code
    › Marked Content (P)
    › Code
    › Marked Content (P)
› H4
› H5
› P
```