developer.mozilla.org

Text labels and names

13-16 minutes

There are many situations in which a control, dialog, or other website feature should be given a descriptive name or label to allow users of assistive technologies to understand what its purpose is and to be able to understand and operate it correctly. There are a number of different types of problems in this category, found in different contexts, and each has its own solution. The different problems and solutions are discussed in the sections below.

Use alt attribute to label area elements that have the href attribute

In image maps, give each `<area>` element an `alt` attribute containing a name that describes what resources the areas links to. Failure to do so makes an image map hard to use for users of assistive technology — they need alternative text to be able to understand the purpose of an image.

Examples

The following example show a simple image map (taken from H24: Providing text alternatives for the area elements of image maps):

```
<img src="welcome.gif" usemap="#map1"
    alt="Areas in the library. Select an area for
more information on that area." />
<map id="map1" name="map1">
  <area shape="rect" coords="0,0,30,30"
    href="reference.html" alt="Reference" />
  <area shape="rect" coords="34,34,100,100"
```

```
    href="media.html" alt="Audio visual lab" />
</map>
```

See the <area> element reference page for a live interactive example.

See also

- <area>
- H24: Providing text alternatives for the area elements of image maps

Dialogs should be labeled

For any container whose contents act as a dialog box (for example, a modal dialog asking the user to make a choice or respond to an action being taken), give it a descriptive label or name, so that assistive technology users can easily discover what its purpose is.

A dialog box is generally denoted by an ARIA role="dialog" or role="alertdialog"; you can use the aria-label or aria-labelledby attributes to provide a label.

Examples

The following example shows a simple dialog box, defined as such using role="dialog" and labelled using aria-labelledby.

```
<div role="dialog" aria-labelledby="dialog1Title"
aria-describedby="dialog1Desc">
  <h2 id="dialog1Title">Your personal details were
successfully updated</h2>
  <p id="dialog1Desc">You can change your details
at any time in the user account section.</p>
  <button>Close</button>
</div>
```

If the dialog box doesn't have a heading, you can instead use aria-label to contain the label text:

```
<div role="dialog" aria-label="Personal details
```

```
updated confirmation">
  <p>Your personal details were successfully
updated. You can
    change your details at any time in the user
account section.</p>
  <button>Close</button>
</div>
```

See also

- role="dialog"
- role="alertdialog"
- aria-label
- aria-labelledby
- WAI-ARIA: dialog role
- Dialog authoring practices

Documents must have a title

It is important in each HTML document to include a <title> that describes the page's purpose. A common navigation technique for users of assistive technology is to infer what content a page contains by reading its title. If the title is not available, they have to navigate the page to determine its content, which can be a time consuming and potentially confusing process.

Examples

The title for the reference article about the <title> element is as follows:

```
<title>&lt;title&gt;: The Document Title element -
HTML: Hypertext Markup Language | MDN</title>
```

Another example might look like so:

```
<title>Fill in your details to register – myGov
services</title>
```

To help the user, you can update the page title value to reflect significant page state changes (such as form validation problems):

```
<title>2 errors — Fill in your details to register
— myGov services</title>
```

See also

- `<title>`

Embedded content must be labeled

Make sure that elements that embed content have a title attribute that describes the embedded content. This includes the `<embed>` and `<object>` elements. These elements are often used for graphical content, much like the `<img>` element. A descriptive title helps users of assistive technology understand what the element is showing.

Figures with optional captions should be labeled

For best accessibility, include a `<figcaption>` within a `<figure>` element, even though doing so is technically optional. The caption is in addition to any alternative text on images within the figure. The caption describes the purpose of the figure in the document, which may be different from a simple description of a visual item, as provided by the alternative text.

Example

The following example shows code for a figure with a caption. The `alt` attribute of the `<img>` describes the appearance of the image; the `<figcaption>` describes it from a functional perspective (in this case, the Latin name of the flower in the image).

```
<figure>
  <img src="milkweed.jgp"
      alt="Black and white close-up photo of
milkweed flowers">
 <figcaption>Asclepias verticillata</figcaption>
```

```
</figure>
```

Fieldset elements must be labeled

Fieldset elements must have a text description, similar to other form elements. Use the `<legend>` element to describe the purpose of a fieldset.

Use a legend to label a fieldset

When grouping a set of form elements together with a `<fieldset>` element, you should include a nested `<legend>` element inside it, containing a clear description of the group.

Users of assistive technology find this description helpful when trying to work out the overall purpose of the group. Without the legend, they would have to navigate around the individual form controls in the group to infer an idea of the overall purpose, which could result in confusion.

Examples

```
<form>
  <fieldset>
    <legend>Choose your favorite monster</legend>

    <input type="radio" id="kraken"
name="monster">
    <label for="kraken">Kraken</label><br/>

    <input type="radio" id="sasquatch"
name="monster">
    <label for="sasquatch">Sasquatch</label><br/>

    <input type="radio" id="mothman"
name="monster">
    <label for="mothman">Mothman</label>
  </fieldset>
</form>
```

You can see a live, interactive version of this example on the `<fieldset>` reference page.

See also

- `<fieldset>`
- `<legend>`

Form elements must be labeled

All elements within a form must have a `<label>` that identifies its purpose. This applies to all types of `<input>` items, as well as `<button>`, `<output>`, `<select>`, `<textarea>`, `<progress>` and `<meter>` elements, as well as any element with the `switch` ARIA role.

The form element can be placed inside the `<label>`, in which case the association between the form element and the label is obvious from the structure. Or, you can create an association between a `<label>` and a form element by specifying the form element's `id` value as the value of the label's `for` attribute.

Example

```
<label>I agree to the terms and conditions.
  <input type="checkbox" id="terms">
</label>

<input type="checkbox" id="emailoptin">
<label for="emailoptin">Yes, please send me news
about this product.</label>
```

Form elements should have a visible text label

In addition to having a `<label>` for every form element, those labels should be visible, not hidden. Visible labels help *all* users understand the purpose of a form element. Do not rely on placeholder text, because it disappears as soon as the user starts typing.

Frame elements must be labeled

Frame elements, both `<iframe>` and the older, obsolete `<frame>`, must have a title to describe the contents of the frame. Use the `title` attribute to label a frame element. Without a title, users of assistive technologies have to navigate into the frame in order to understand what it contains, which can be difficult and confusing.

The `<frame>` element is no longer part of the HTML specification as of HTML5. Support for it may be dropped by browsers in the future. In addition, it is difficult for screen readers to navigate pages with `<frame>` elements. For best accessibility and future maintenance, redesign any pages that use frames to use CSS to achieve a similar layout.

As a best practice, also provide a `<title>` for the document that is enclosed in the frame, with content identical to the frame's `title` attribute. (This assumes that the enclosed document is under your control; if not, try to match the frame's `title` attribute to the document's title.) Some screen readers replace the contents of the `title` attribute with the contents of the enclosed document's `<title>`. It's safest and most accessible to provide the same title in both places.

Example

```
<iframe
    title="MDN Web docs"
    width="300"
    height="200"
    src="https://developer.mozilla.org">
</iframe>
```

Use alt attribute to label mglyph elements

When writing equations with MathML, give each `<mglyph>` element an `alt` attribute containing a name that describes the symbol. Since `mglyph` elements are used for non-standard

symbols without Unicode definitions, screen readers won't automatically be able to name them. Alternative text helps users of screen readers understand the symbol.

Headings must be labeled

Make sure that your headings have non-empty text content, and are not hidden, such as with CSS `display:none` or `aria-hidden=true`. Users of screen readers rely on headings to understand the structure and content of a document.

Also, be sure you are using heading elements only for actual section headings, and not as a shortcut way to make text stand out. Screen reader users typically "skim" a page's headings, much like sighted users; non-heading text that is marked-up with heading elements can cause confusion.

Headings should have visible text content

Make sure that your headings have non-empty text content, and are not hidden, such as with CSS `display:none` or `aria-hidden=true`. Users of screen readers rely on headings to understand the structure and content of a document. Do not use heading elements to mark up images or other graphical content.

Use title attribute to describe <iframe> content

Make sure that `<iframe>` elements have a `title` attribute to describe the contents of the frame. Without a title, users of assistive technologies have to navigate into the frame in order to understand what it contains, which can be difficult and confusing.

As a best practice, also provide a `<title>` for the document that is enclosed in the frame, with content identical to the frame's `title` attribute. (This assumes that the enclosed document is under your control; if not, try to match the frame's `title` attribute to the document's title.) Some screen readers replace the contents of the `title` attribute with the contents of the enclosed document's `<title>`. It's safest and most accessible to provide the same title

in both places.

Content with images must be labeled

Provide descriptive text for all contentful (that is, non-decorative) images and image-like elements. This includes SVG images, `<img>`, `<canvas>`, `<map>`, and `<area>` elements, as well as `<input>` elements where `type=image` and `<object>` elements where the `type` starts with `image/`. The typical way to do this is with the `alt` attribute. Be sure that the description conveys what is shown in the image.

Example

```
<img src="milkweed.jgp"
     alt="Black and white close-up photo of
milkweed flowers">
```

Interactive elements must be labeled

If an element is intended for users to interact with it, it should have a label. Interactive elements include links (`<a>`), form elements, buttons, and any element that has a handler for mouse or keyboard events. The way to label an element depends on its type: for form elements, use a `<label>`; for links, buttons and clickable elements, the text content of the element typically provides the label. If no other option exists for labeling an element, use the `aria-label` attribute.

Use label attribute on optgroup elements

In an `<optgroup>` element, use the `label` attribute to describe the group so that assistive technologies can access it for their users.

Example

In this example, the `label` attribute on the `<optgroup>` elements gives a category name for the group of options.

```
<label for="dino-select">Choose a dinosaur:
</label>
<select id="dino-select">
    <optgroup label="Theropods">
        <option>Tyrannosaurus</option>
        <option>Velociraptor</option>
        <option>Deinonychus</option>
    </optgroup>
    <optgroup label="Sauropods">
        <option>Diplodocus</option>
        <option>Saltasaurus</option>
        <option>Apatosaurus</option>
    </optgroup>
</select>
```

If you define more than one toolbar in a web application using the ARIA `toolbar` role, you must use the `aria-label` attribute to label each one so that it can be described by assistive technology. It is a good practice to label a toolbar, even if there is only one per page.

See also

- W3C ARIA toolbar example

1.1.1 Non-text Content (A)
: All non-text content that is presented to the user has a text alternative that serves the equivalent purpose, except for the situations listed in the above link.

2.4.4 Link Purpose (In Context) (A)
: The purpose of each link can be determined from the link text alone or from the link text together with its programmatically determined link context, except where the purpose of the link would be ambiguous to users in general.

2.4.9 Link Purpose (Link Only) (AAA)
: A mechanism is available to allow the purpose of each link to be identified from link text alone, except where the purpose of the link would be ambiguous to users in general.