AcceDe Web

ATALAN
eAccessibility consultancy
Accessibilité numérique

Accessibility guidelines for HTML and CSS

Date	Version	Author	Statuts / comments
Nov. 3rd, 2018	3.0	Atalan	
Jan 21, 2019	3.1	Atalan	Minor changes in text of the 6.2 guideline.
August 10, 2020	4.0	Atalan	Major updates to ensure compliance with RGAA 4 and WCAG 2.1 (levels A and AA).

In partnership with :
Air Liquide – Atos – BNP Paribas – Capgemini – EDF – Generali – L'Oréal – SFR – SNCF – Société Générale – SPIE – Total

Observers :
AbilityNet *(UK)* – Agence Entreprises & Handicap – AnySurfer *(Belgium)* – Association des Paralysés de France (APF) – Association Valentin Haüy (AVH) – CIGREF – Design For All Foundation (Spain) – ESSEC – Handirect – Hanploi – Sciences Po – Télécom ParisTech

Table of contents

TABLE OF CONTENTS **2**

INTRODUCTION **6**

Context and objectives 6

Who should read this document, and how to use it? 6

Contact 6

License agreement 7

1. GENERAL STRUCTURE **8**

1.1 Add structure to the header area using `<header role="banner">` 8

1.2 Mark up the search field with `role="search"` 8

1.3 Structure the main content area with `<main role="main">` 8

1.4 Structure information about the website with `<footer role="contentinfo">` 9

1.5 Structure primary and secondary navigation menus with `<nav role="navigation">` 9

1.6 Structure navigation menus with lists 10

Why should I structure lists? *10*

1.7 Create a logical and thorough hierarchy for headings, using `<h1>` to `<h6>` tags 11

2. PAGE TITLE **16**

2.1 Provide a descriptive `<title>` element on each page 16

3. LANGUAGES **17**

3.1 Provide the primary language of the page with the `lang` attribute on the `<html>` tag 17

3.2 Use the `lang` attribute to indicate a change of language 17

4. HTML GRAMMAR AND SEMANTICS **19**

4.1 Write a valid HTML code following the formal grammar rules of the DOCTYPE used 19

4.2 Use HTML elements for their semantic purpose 19

5. LINKS AND BUTTONS **20**

5.1 Distinguish buttons from links 20

Buttons *20*

Links *20*

5.2 Complete non-explicit links and buttons using hidden text, `aria-label` or `title`. 20

The `aria-label` attribute ... 20

The `title` attribute ... 21

5.3 Don't use the `aria-label` or `title` attributes on links and buttons that are already explicit ... 22

6. IMAGES AND ICONS ... 23

6.1 Managing the alt attribute of the `<img />` and `<input type="image" />` tags ... 23

Decorative/ambient `<img />` ... 23

"Simple" informative `<img />` ... 23

"Complex" informative `<img />` ... 24

Unlabeled `<img />` links or buttons ... 25

`<input type="image" />` ... 26

6.2 Managing alternative text on SVG (Scalable Vector Graphics) ... 26

Decorative/ambient `<svg>` ... 26

"Simple" informative `<svg>` ... 27

"Complex" informative `<svg>` ... 28

`<svg>` links and buttons alone ... 29

6.3 Managing alternative text on Icon Fonts ... 30

Decorative/ambient icons ... 30

Informative icons ... 30

Links or buttons icons ... 31

6.4 Tag images that have captions with `<figure role="figure">` and `<figcaption>` ... 32

Example 1 ... 32

Example 2 ... 33

7. FORMS ... 34

7.1 Use the `<label>` tag with the for and `id` attributes to label form fields with a visible name ... 34

7.2 Use the `title` attribute to label form fields that don't have a visible label ... 34

7.3 Use the `autocomplete` attribute to facilitate auto-fill ... 35

7.4 Add help text directly to the `<label>` tags ... 35

7.5 Add `required` or `aria-required="true"` to mandatory fields ... 37

7.6 Add error messages and correction suggestions directly to the `<label>` tag ... 37

7.7 Group and label related fields with `<fieldset>` and `<legend>` ... 39

8. LISTS 41

8.1 Tag unordered lists with `<ul>` and `<li>` 41

What are the advantages? 41

8.2 Tag ordered lists with `<ol>` and `<li>` 41

What are the advantages? 42

8.3 Mark up description lists with `<dl>`, `<dt>` and `<dd>` 42

What are the advantages? 43

9. TABLES 44

9.1 Use `<caption>` to tag data table captions 44

9.2 Mark up the row and column header cells with `<th>` 44

9.3 In simple data tables use the `scope` attribute to associate data cells with their corresponding headers 45

9.4 Add `role="presentation"` to each layout `<table>` 46

10. USING CSS 47

10.1 Use CSS to lay out text 47

10.2 Ensure that content can be read when images are turned off 47

10.3 Ensure that content is understandable when CSS is disabled 48

Examples 48

11. ZOOM AND FONT SIZES 51

11.1 Use only relative font sizes (`rem`, `em`, `%`, etc.) 51

11.2 Ensure content readability even when font size is doubled 51

12. KEYBOARD NAVIGATION 52

12.1 Ensure the proper functioning of the interface using either the mouse or keyboard 52

12.2 Ensure that the tab order follows the logical reading order 52

12.3 Ensure that keyboard focus is visible 53

12.4 Introduce “skip links” 53

13. IFRAMES 55

13.1 Populate the title attribute on each `<iframe` 55

14. ADDITIONAL GUIDELINES 56

Tag block quotations using `<blockquote>` 56

Tag online quotations using `<q>` 56

Write the HTML code in a logical reading order 57

Indicate in the alternative text for visual CAPTCHA where to find the non-visual version .. 57
Do not use tags or attributes specific to data tables in layout table 58
Plan alternative text for each type of multimedia content (`<video>`, `<audio>`, `<object>`, `<embed>`, etc.) 58
Populate the `alt` attribute of each mapped image and its `<area />` tags 59
In complex tables, use the `headers` and `id` attributes to associate header cells with their corresponding data cells 59
Ensure that the HTML flow is consistent from one page to the next 60
Make sure layout tables follow the right reading order 61
ADDITIONAL RECOMMENDATIONS 63
Tag abbreviations with `<abbr>` 63
Combine adjacent links pointing to the same page 63
In forms with multiple steps, identify the current step using `aria-current="step"` 64
Identify the current position in navigation systems with `aria-current` 65
Example in a menu 65
Example in a pagination system 66
Example in a breadcrumb trail 66
Add file format and size to all Download links and buttons 66
Update the page `<title>` when an error or confirmation message is displayed 67
Do not disable the zoom with the `user-scalable` property 68
Do not use CSS to display images that carry information 68
Logically organize the items of dropdown lists 69
Send a notification when a new window is opened 70
Use only relative sizes (`rem`, `em`, `%`, etc.) for media query breakpoints 71
Make sure that hidden content is translated 71

Introduction

Context and objectives

This document provides guidelines for making HTML and CSS integration more accessible.

This manual is part of a set of four complementary manuals that can be downloaded from the AcceDe Web website at www.accede-web.com:

- Graphic and functional accessibility guidelines.
- **HTML and CSS accessibility guidelines (this manual).**
- Main rich interface components accessibility guidelines.
- Accessibility guidelines for editors (template).

Who should read this document, and how to use it?

This document should be provided to stakeholders and/or contractors implementing the technical specifications, and HTML/CSS integration. It complements project specifications. The recommendations may be supplemented or removed depending on the context of use, such work can be carried out by the project owner.

These recommendations must be used when creating HTML/CSS. Some of them are used for dynamic content, for example, when templates are created for a content management system (CMS).

Note

The online version of these guidelines comes with many examples, links to complementary resources, etc. It is available at: www.accede-web.com/en/.

Contact

Please send any comments about this document to Atalan, the initiator of the AcceDe Web project, at the following email address: accede@atalan.ca.

You can also find more information on the Guidelines methodology of the AcceDe Web project on the website www.accede-web.com.

1. General structure

1.1 Add structure to the header area using `<header role="banner">`

Mark up the main header section (i.e. logo, search field, etc.) with `<header role="banner">`.

Warning

While the `<header>` tag may be used several times in a web page, `role="banner"` must only be used once per web page.

Note

It is perfectly acceptable to nest several ARIA roles: `<div role="search">` into `<header role="banner">`, for example.

```
<header role="banner">
   <img src=" my-site-logo.png" alt="My site (logo" />
   <div role="search" … > […] </div>
   […]
</header>
```

1.2 Mark up the search field with `role="search"`

The website search field must be identified with `role="search"`.

```
  <form role="search" … >
    <input type="text" title="Keyword Search" />
    <input type="submit" value="Search" />
  </form>
```

1.3 Structure the main content area with `<main role="main">`

The main content area of the page must be marked up with `<main role="main">`.

Warning

A page must contain only one **visible** `<main role="main">` tag.

```
<header role="banner">[…]</header>
<nav role="navigation">[…]</nav>
<main role="main">[…]</main>
<footer role="contentinfo">[…]</footer>
```

1.4 Structure information about the website with `<footer role="contentinfo">`

Information about the website such as the copyright and the “Legal terms and conditions” and “Credits” links must be structured with `<footer role="contentinfo">`.

Warning

A page can contain only one `role="contentinfo"`.

```
<header role="banner">[…]</header>
<nav role="navigation">[…]</nav>
<main role="main">[…]</main>
<footer role="contentinfo">
   <ul>
      <li><a href="…">Sitemap</a></li>
      <li><a href="…">Credits</a></li>
      <li><a href="…">Terms and Conditions</a></li>
   </ul>
   <p>© Company/Site name</p>
</footer>
```

1.5 Structure primary and secondary navigation menus with `<nav role="navigation">`

Primary and secondary navigation menus must be marked up with `<nav role="navigation">`.

Note

The `<nav>` tag is reserved for website’s **internal links**.

For example:

- The main menu.
- The sub-menu.
- The breadcrumb.
- The search results page number.

In other words, it must not be used to mark up a list of external links, such as a list of social media links.

Tip

Good practice is to add an `aria-label` attribute to the `<nav role="navigation">` tag to specify the type of navigation system.

```
<header role="banner">[…]</header>
<nav role="navigation" aria-label="Primary menu">[…]</nav>
<nav role="navigation" aria-label="Secondary menu">[…]</nav>
<main role="main">
   <nav role="navigation" aria-label="You are here">[…]</nav>
   […]
</main>
<footer role="contentinfo">[…]</footer>
```

1.6 Structure navigation menus with lists

Use unordered list elements `<ul>` and `<li>` to mark up navigation menus.

For primary menus with several levels, be sure to nest the links appropriately:

```
<nav role="navigation" aria-label="Main menu">
 <ul>
  <li><a href="…">First menu item</a></li>
  <li>
    <a href="…">Second menu item</a>
    <ul>
      <li><a href="…">First sub-menu item</a></li>
      <li><a href="…">Second sub-menu item</a></li>
    </ul>
  </li>
  <li><a href="…">Third menu item</a></li>
 </ul>
</nav>
```

Why should I structure lists?

- Structuring lists is important for people who use screen readers (blind and visually-impaired people). When readers come across a list, structuring enables them to:
- Navigate from list to list on a page.
- Know, from the start, the number of items on the page.
- Navigate more easily through the list:
    - Go directly to the end of the list if the content does not interest them.
    - Easily go back to the beginning of the list.

1.7 Create a logical and thorough hierarchy for headings, using `<h1>` to `<h6>` tags

On each page, headings must be tagged from `<h1>` to `<h6>`. The structure of the headings must be both logical and thorough.

In other words:

- There must not be any "jumps" or inconsistencies in the heading structure (i.e. from `<h1>` to `<h3>` without `<h2>` in between).
- All elements that function as headings must be marked up using heading tags.

Notes

- Several `<h1>` elements can be used on a page if there are primary headings.
- Good accessibility practice is to avoid using hidden headings.

Tip

To create a logical and understandable heading hierarchy, imagine the headings of the page as a "Table of Contents". Is it logical? thorough?

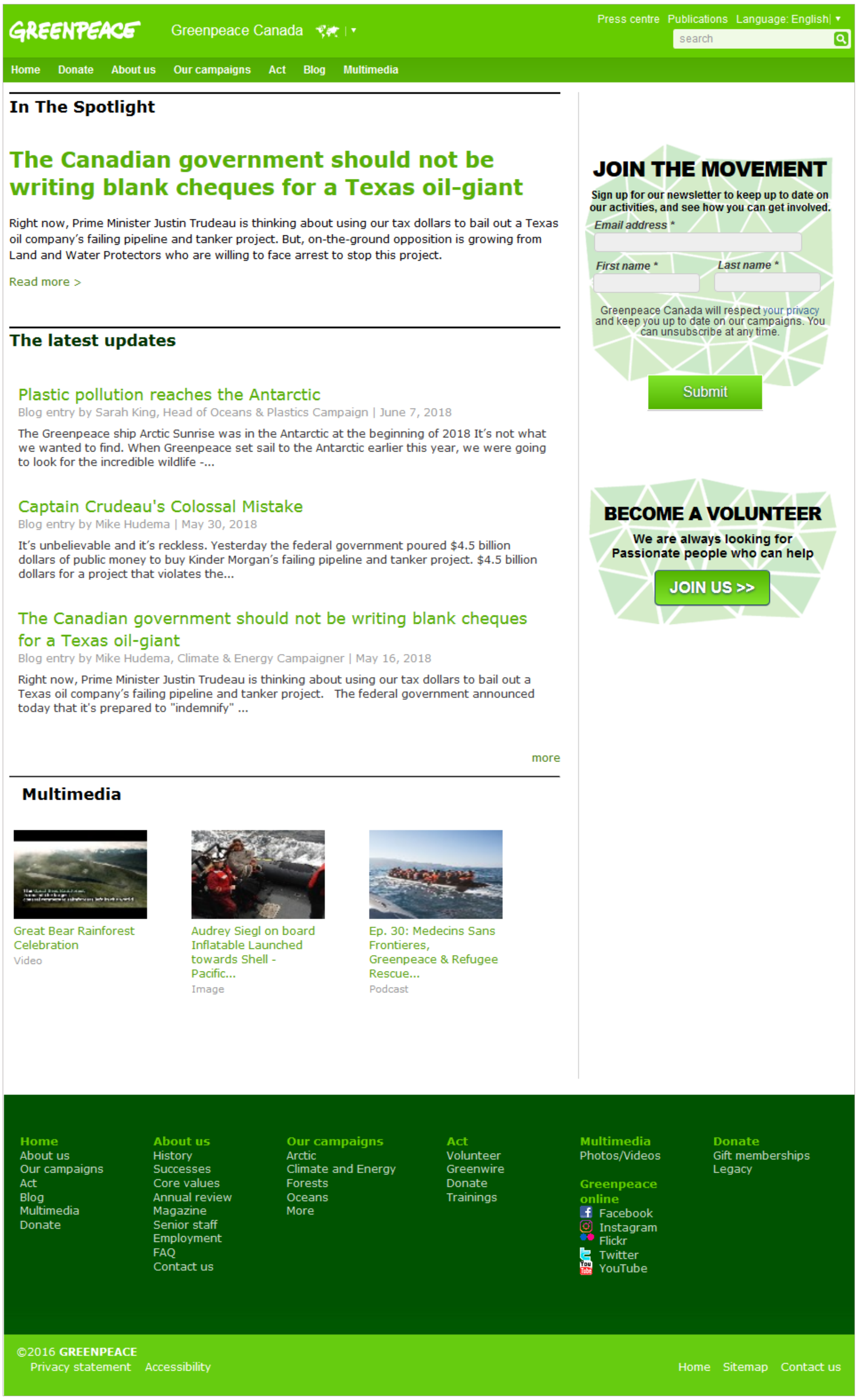
Note

The ARIA `heading` role with the `aria-level` attribute (with a value of 1 to 6) will override the semantics of any native HTML tag/ makes it possible to assign a title value to any HTML tag.

For example:

- `<p role="heading" aria-level="1">Heading Level 1</p>` is semantically equivalent to `<h1>Heading Level 1</h1>`.
- `<div role="heading" aria-level="3">Heading Level 3</div>` is semantically equivalent to `<h3>Heading Level 3</h3>`.

However, this technique is not optimal for accessibility, it should only be used as a last resort.



GREENPEACE Greenpeace Canada

Press centre Publications Language: English|

search

Home Donate About us Our campaigns Act Blog Multimedia

In The Spotlight

The Canadian government should not be writing blank cheques for a Texas oil-giant

Right now, Prime Minister Justin Trudeau is thinking about using our tax dollars to bail out a Texas oil company's failing pipeline and tanker project. But, on-the-ground opposition is growing from Land and Water Protectors who are willing to face arrest to stop this project.

Read more >

The latest updates

Plastic pollution reaches the Antarctic
Blog entry by Sarah King, Head of Oceans & Plastics Campaign | June 7, 2018

The Greenpeace ship Arctic Sunrise was in the Antarctic at the beginning of 2018 It's not what we wanted to find. When Greenpeace set sail to the Antarctic earlier this year, we were going to look for the incredible wildlife -...

Captain Crudeau's Colossal Mistake
Blog entry by Mike Hudema | May 30, 2018

It's unbelievable and it's reckless. Yesterday the federal government poured $4.5 billion dollars of public money to buy Kinder Morgan's failing pipeline and tanker project. $4.5 billion dollars for a project that violates the...

The Canadian government should not be writing blank cheques for a Texas oil-giant
Blog entry by Mike Hudema, Climate & Energy Campaigner | May 16, 2018

Right now, Prime Minister Justin Trudeau is thinking about using our tax dollars to bail out a Texas oil company's failing pipeline and tanker project. The federal government announced today that it's prepared to "indemnify" ...

more

Multimedia

Great Bear Rainforest Celebration
Video

Audrey Siegl on board Inflatable Launched towards Shell - Pacific...
Image

Ep. 30: Medecins Sans Frontieres, Greenpeace & Refugee Rescue...
Podcast

JOIN THE MOVEMENT

Sign up for our newsletter to keep up to date on our activities, and see how you can get involved.

Email address *

First name * Last name *

Greenpeace Canada will respect your privacy and keep you up to date on our campaigns. You can unsubscribe at any time.

Submit

BECOME A VOLUNTEER

We are always looking for Passionate people who can help

JOIN US >>

Home	About us	Our campaigns	Act	Multimedia	Donate
About us	History	Arctic	Volunteer	Photos/Videos	Gift memberships
Our campaigns	Successes	Climate and Energy	Greenwire		Legacy
Act	Core values	Forests	Donate	Greenpeace online	
Blog	Annual review	Oceans	Trainings	Facebook	
Multimedia	Magazine	More		Instagram	
Donate	Senior staff			Flickr	
	Employment			Twitter	
	FAQ			YouTube	
	Contact us				

©2016 GREENPEACE
Privacy statement Accessibility

Home Sitemap Contact us

Three examples of heading structures for this page are provided below. The first two are correct, but the last one is not.

Example 1 (correct)

```
<h1><a href="/"><img src="GreenPeace.png" alt="GreenPeace (go
back to homepage)" /></a></h1>
[…]
<h2>In the spotlight</h2>
[…]
<h3>The Canadian government should not be writing blank
cheques for Texas oil-giant</h3>
[…]
<h2>The latest updates</h2>
[…]
<h3>Plastic pollution reaches the Antarctic</h3>
[…]
<h3>Captain Crudeau’s Colossal Mistake</h3>
[…]
<h3>The Canadian government should not be writing blank
cheques for Texas oil-giant</h3>
[…]
<h2>Multimedia</h2>
[…]
<h2>Join the movement</h2>
[…]
<h2>Become volunteer</h2>
[…]
```

In this example of HTML code, the heading structure is logical and thorough

Example 2 (correct)

```
<a href="/"><img src="GreenPeace.png" alt="GreenPeace (go back
to homepage)" /></a>
[…]
<h1>In the spotlight</h1>
[…]
<h2>The Canadian government should not be writing blank
cheques for Texas oil-giant</h2>
[…]
<h1>The latest updates</h1>
[…]
<h2>Plastic pollution reaches the Antarctic</h2>
[…]
<h2>Captain Crudeau’s Colossal Mistake</h2>
[…]
<h2>The Canadian government should not be writing blank
cheques for Texas oil-giant</h2>
[…]
```

```
<h1>Multimedia</h1>
[…]
<h1>Join the movement</h1>
[…]
<h1>Become volunteer</h1>
[…]
```

In this example of HTML code, the heading structure is also logical and thorough.

Note that other heading structures are possible.

Example 3 (incorrect)

```
<a href="/"><img src="GreenPeace.png" alt="GreenPeace (go back
to homepage)" /></a>
[…]
<h1>In the spotlight</h1>
[…]
<h2>The Canadian government should not be writing blank
cheques for Texas oil-giant</h2>
[…]
<h1>The latest updates</h1>
[…]
<h4>Plastic pollution reaches the Antarctic</h4>
[…]
<h4>Captain Crudeau's Colossal Mistake</h4>
[…]
<h4>The Canadian government should not be writing blank
cheques for Texas oil-giant</h4>
[…]
<h1>Multimedia</h1>
[…]
<p>Join the movement</p>
[…]
<p>Become volunteer</p>
[…]
```

In this example of HTML code, the heading structure is incorrect because it includes “jumps” and inconsistencies (sharp transition from `<h1>` to `<h4>`).

Also, “Join the movement” and “Become a volunteer” are not tagged as a heading.

2. Page title

2.1 Provide a descriptive `<title>` element on each page

On each page, the `<title>` element must include a descriptive title. It must, at least, provide both the name of the page and the name of the website.

Tip

- The name of the current page should come first in the `<title>` element before other information. For example, `<title>[Current page name] | [Website name]</title>`.
- Good practice is to ensure that the content in the `<title>` element is consistent throughout the entire website.

Note

In certain circumstances, the content is modified when the page is reloaded. This may be the case, for example:

- When pagination is used.
- When a search term is entered into a search field.

The title of the affected page must be modified. For example:

```
<title>Search results for [Search term] (page 3/7) | [Site
name]</title>
```

3. Languages

3.1 Provide the primary language of the page with the `lang` attribute on the `<html>` tag

In order to ensure good rendering of text content, the primary language must be declared on each page.

Use the `lang` attribute on the `<html>` tag.

For example, for an English page use:

```
<html lang="en">
```

Tip

To populate the `lang` attribute, a two-letter language code (or, if it is not available, three-letter code) must be used.

Some common codes are:

- `en` for English.
- `fr` for French.
- `es` for Spanish.
- `de` for German.
- `it` for Italian.

The others are available here: exhaustive list of authorized language codes.

3.2 Use the `lang` attribute to indicate a change of language

If some of the content is in a language other than the primary language of the page, it must be identified using the `lang` attribute.

For example, on an English page:

```
<a href="…" lang="fr" hreflang="fr">French version</a>
```

Tip

If the foreign language content is not tagged directly, use a `<span>` or `<div>` tag and populate the `lang` attribute.

Note

It is not necessary to announce a change of language for:

- Proper names
- Foreign words that have been incorporated into the dictionary of the primary language.
- Foreign words that are pronounced and fully understood with the accent of the main language (i.e. “RSVP”, if the primary language is English).

4. HTML Grammar and Semantics

4.1 Write a valid HTML code following the formal grammar rules of the DOCTYPE used

Use a valid DOCTYPE for each page and make sure the **generated** HTML code complies with the formal grammar rules of that DOCTYPE (the choice of DOCTYPE is open).

Note

This DOCTYPE must be placed before the `<html>` tag.

The following aspects of formal grammar are particularly important:

- Elements are nested according to their specifications.
- Elements have complete start and end tags
- There are no duplicate attributes on an element.
- Each id attribute has a unique value on the page.

Warning

Obsolete HTML elements and attributes and/or those intended for formatting purposes only must not be used.

4.2 Use HTML elements for their semantic purpose

HTML elements must be used for their semantic purpose and not solely for the visual aspect of content.

Tags	Use it to	Don't use it to
<a>	Mark up a hypertext link	Create an action button with no background or border
<hr />	Create a semantic division between distinct groups of content in a text box.	Create a visual separator line
<fieldset>	Group form elements which are related to one another	Create a visual border
<q>	Mark up a short quote	Place quotation marks around non-quoted text

5. Links and Buttons

5.1 Distinguish buttons from links

Limit the use of buttons to actions and links to navigation.

Buttons

Button elements `<button />` / `<input />` are for triggering an action.

They are to be used in cases such as submitting information, displaying the next or previous image in a carousel, closing a modal dialogue box, etc.

Links

Links `<a>` are for navigation.

They are used to move to another page or to a specific area of the current page, by means of an anchor system.

5.2 Complete non-explicit links and buttons using hidden text, `aria-label` or `title`.

A link or button is non-explicit when the label alone does not suffice to describe its destination or function.

- Link labels such as "Read More", "Learn More", "More Information", "Click Here" are considered as non-explicit by nature.
- So is the "OK" button.

In these cases, the `aria-label` attribute or the `title` attribute must be used to specify the destination of the link or the function of the button.

As an introduction to these two techniques, let us consider the example below of a “Read more” link pointing to a page of the AcceDe Web project.

The `aria-label` attribute

A link or button can be made more explicit using the `aria-label`:

1. Add the `aria-label` attribute to the `<a>`, `<button>` or `<input />` tag.
2. Populate this attribute by entering the exact label of the link or button followed by the information to make the link or button explicit.

```
<a href="…" aria-label="Read more: The AcceDe Web project">
   Read more
</a>
```

The `title` attribute

Remarque

The `title` attribute is only partly supported by browsers and assistive technologies.

Although it is an accepted technique, it is preferable to use the `aria-label` attribute.

A link or button can be made more explicit using the `title` attribute:

1. Add the title attribute to the `<a>`, `<button>` or `<input />` tag.
2. Populate this attribute by entering the exact label of the link or button followed by the information to make the link or button explicit.

```
<a href="…" title="Read more: The AcceDe Web project">
   Read more
</a>
```

Tip

Good practice is to populate these attributes by first entering the label then providing the necessary information to make the link or button more explicit.

Note

One of these two techniques must also be used to distinguish links or buttons whose labels might be considered as explicit but which lead to different pages or trigger different actions.

If there are two “Search” buttons on the same page, for example, they need to be distinguished:

```
<input type="submit" value="Search" aria-label="Search a news"
/>
[…]
<input type="submit" value="Search" aria-label="Search a
person in the directory" />
```

Note

Note that a link can be considered as explicit though its context when the surrounding elements can help understand its meaning.

5.3 Don't use the `aria-label` or `title` attributes on links and buttons that are already explicit

A link or button is explicit when the label alone and/or the context suffice to describe its destination or function.

For example:

- The link "AcceDe Web Project" is explicit by nature.
- So is a button labelled "Confirm Order".

The `aria-label` and `title` attributes (which can be used to make non-explicit links accessible) must not be used for this type of link.

Here are some examples of **incorrect** use of the `aria-label` on an `<input />` tag:

✗
```
<input type="submit" value="Confirm Order" aria-label="" />
```

✗
```
<input type="submit" value="Confirm Order" aria-label="Confirm
my Order" />
```

And here are some examples of **incorrect** use of the `title` attribute on an `<a>` tag:

✗
```
<a href="…" title="">AcceDe Web project</a>
```

✗
```
<a href="…" title="AcceDe Web">AcceDe Web project</a>
```

6. Images and Icons

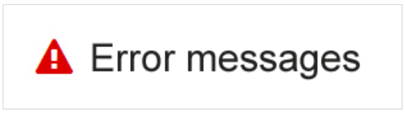
6.1 Managing the alt attribute of the `<img />` and `<input type="image" />` tags

5 different cases:

- Decorative/ambient `<img />`.
- “Simple” informative `<img />`.
- “Complex” informative `<img />`.
- Unabeled `<img />` links or buttons.
- `<input type="image" />`.

Decorative/ambient `<img />`

When a decorative or ambient `<img />` tag is included in the HTML code, the `alt` attribute must remain empty (without any space between the quotation marks of `alt=""`).



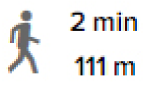
In the example of HTML code below, the `<img />` tag simply enhances the title "Error messages" (which is explicit by nature):

```
<h2>
   <img src="error.png" alt="" />
   Error messages
</h2>
```

“Simple” informative `<img />`

A “simple” informative image is one that can be described by concise alternative text.

When a simple informative `<img />` tag is used in the HTML code, populate its `alt` attribute with the information conveyed by the image.



In the example of HTML code below, the `<img />` tag refers to a walking time and distance:

```
<p>
  <img src="walk.png" alt="By walking:" />
  <span>2 min</span>
  <span>111 m</span>
</p>
```

Warning

Do not start alternative text with `alt="Image of […]"`.

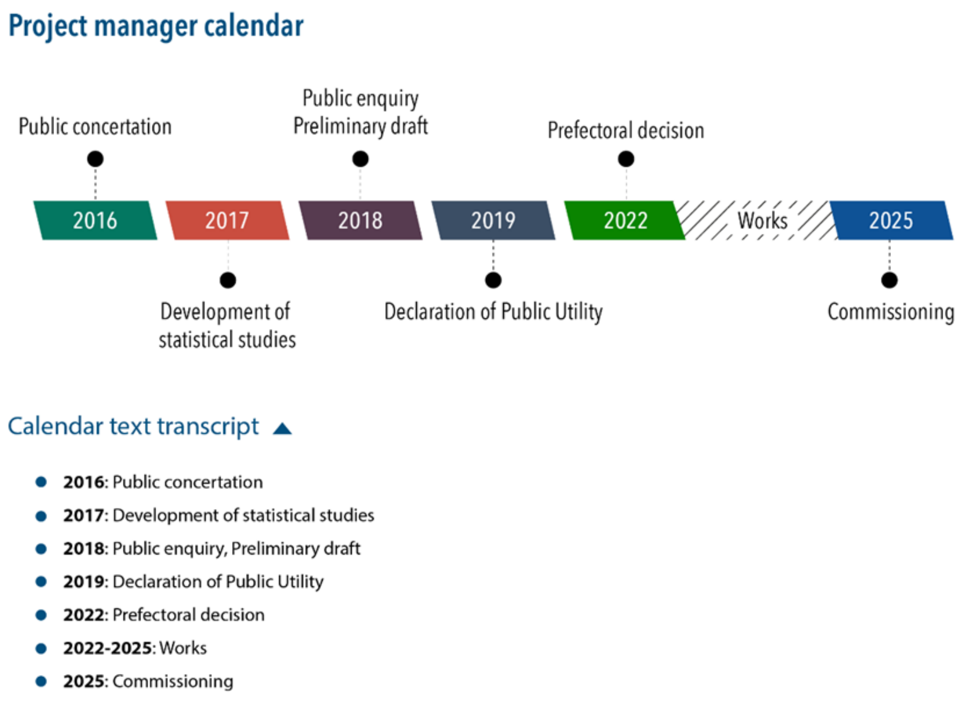
This information is already provided by the assistive technology when it reads the `<img />` tag.

"Complex" informative `<img />`

A "complex" informative image is one that requires a **detailed** description.

When a complex informative `<img />` tag is included in the HTML code:

1. Populate its `alt` attribute with information to make the image explicit.
2. Propose a detailed description of the image directly below it.
3. Lastly, indicate in the `alt` attribute where the detailed description can be found.



- **2016**: Public concertation
- **2017**: Development of statistical studies
- **2018**: Public enquiry, Preliminary draft
- **2019**: Declaration of Public Utility
- **2022**: Prefectoral decision
- **2022-2025**: Works
- **2025**: Commissioning

In this example, a show/hide button displays a detailed description of the complex image.

In the following example of HTML code, the `<img />` tag indicates both the function of the image and where to find its detailed description:

```
<h2>Project Manager calendar</h2>
<img src="female-reproductive-system.png" alt="Project manager
calendar (detailed description below)" />
<button aria-expanded="true">Calendar text transcript</button>
<div>
  <ol>
    <li><strong>2016:</strong> Public concertation</li>
    […]
  </ol>
</div>
```

⚠ **Warning**

Do not start alternative text with `alt="Image of […]"`.

This information is already provided by the assistive technology when it reads the `<img />` tag.

Unlabeled `<img />` links or buttons

When an `<img />` tag acting as link or button is included alone (without a label) in the HTML code, add the information needed to make it explicit into the `alt` attribute.

In the HTML code example below, the `<img />` tag points to the home page:

```
<a href="/">
   <img src="home.png" alt="Home" />
</a>
```

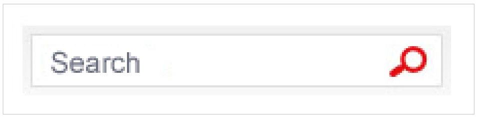
⚠ **Warning**

Do not start the `alt` text with `alt="Link to […]"` or `alt="Button […]"` when the `<img>` tag acts like a link or button.

This information is already provided to assistive technology by the `<a>` or `<button>` tag.

`<input type="image" />`

When `<input type="image" />` is included in the HTML code, add the information needed to make the button explicit into its `alt` attribute.



In the HTML code example below, the `<input type="image" … />` tag launches a search within the website:

```
<input type="image" src="magnifier.png" alt="Search site" … />
```

⚠ **Warning**

Do not start the `alt` attribute with `alt="Button […]"` when the `<input type="image" … />` tag is acting as a link or button.

This information is already provided to the assistive technology by the `<input type="image" … />` tag.

6.2 Managing alternative text on SVG (Scalable Vector Graphics)

4 cases:

- Decorative/ ambient `<svg>`.
- "Simple" informative `<svg>`.
- "Complex" informative `<svg>`.
- Unlabeled `<svg>` links or buttons.

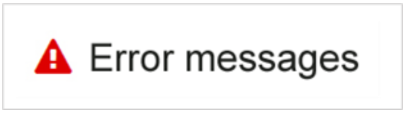
⚠ **Warning**

In Internet Explorer, `<svg>` tags receive keyboard focus by default.

To optimize keyboard navigation in that browser, the `focusable="false"` attribute must be added to each `<svg>` tag used to display an image.

Decorative/ambient `<svg>`

When an `<svg>` is simply decorative or ambient, add `aria-hidden="true"`.



In the following HTML code, the `<svg>` simply enhances the label "Error Messages" (explicit by nature):

```
<h2>
   <svg aria-hidden="true" focusable="false">[...]</svg>
   Error Messages
</h2>
```

⚠ **Warning**

A purely decorative or ambient `<svg>` tag must not have any `title, aria-label`, `aria-labelledby` and/or `aria-describedby` attributes.

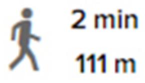
Likewise, it must not have a `<title>` and/or `<desc>` elements.

"Simple" informative <svg>

A "simple" informative `<svg>` is an informative image that can be described by concise alternative text.

When an informative `<svg>` image is included in the HTML code:

1. Add a `role="img"` attribute to the `<svg>` tag.
2. Add an `aria-label` attribute to the `<svg>` tag and populate the attribute with information to make the image explicit.



In the example of HTML code below, the `<svg>` indicates that the text applies to a walking time and distance:

```
<p>
  <svg role="img" aria-label="By walking: " focusable="false">
    [...]
  </svg>
  <span>2 min</span>
  <span>111 m</span>
</p>
```

⚠ **Warning**

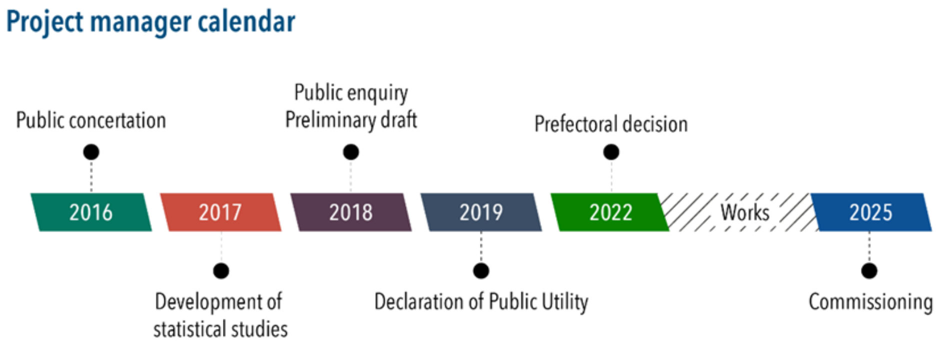
Do not start the `<svg>` alternative text with "Image [...]".

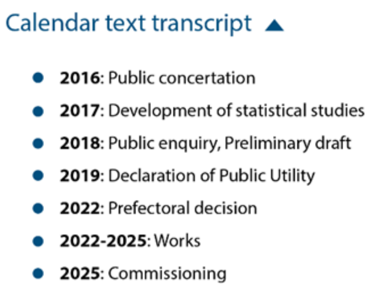
“Complex” informative `<svg>`

A “complex” informative `<svg>` is an informative image that requires a detailed description.

When a “complex” informative `<svg>` is added to the HTML code:

1. Add a `role="img"` attribute to the `<svg>` tag.
2. Add an `aria-label` attribute to the `<svg>` tag and populate this attribute with information to make the image explicit.
3. Propose a detailed description for the image directly below it.
4. Finally, indicate in the `aria-label` attribute where this detailed description can be found.




Calendar text transcript ▲

- **2016**: Public concertation
- **2017**: Development of statistical studies
- **2018**: Public enquiry, Preliminary draft
- **2019**: Declaration of Public Utility
- **2022**: Prefectoral decision
- **2022-2025**: Works
- **2025**: Commissioning

In this example, a show/hide button displays a detailed description of the complex image.

In the example of HTML code below, `<svg>` indicates the function of the image and where to find its detailed description:

```
<h2>Project Manager calendar</h2>
<svg role="img" aria-label="Projet Manager calendar (detailed
description below)" focusable="false">
  [...]
</svg>
<button aria-expanded="true">Calendar text transcript</button>
<div>
  <ol>
```

```
    <li><strong>2016:</strong> Public concertation</li>
    […]
  </ol>
</div>
```

⚠ **Warning**

Do not start the `<svg>` alternative text with "Image […]".

`<svg>` links and buttons alone

When an `<svg>` image alone (without a label), acting as a link or a button, is included in the HTML code:

1. Add a `role="img"` attribute to the `<svg>` tag.
2. Add an `aria-label` attribute to the `<svg>` tag and populate this attribute with information to make the link or button explicit.

In the following example of HTML code, the `<svg>` points to the homepage of a website:

```
<a href="/">
  <svg role="img" aria-label="Homepage" focusable="false">
    [...]
   </svg>
</a>
```

⚠ **Warning**

Do not start the `<svg>` alternative text of a link or a button with “Link to […]” or “Button to […]”.

6.3 Managing alternative text on Icon Fonts

Note

Whether the icon is decorative, informative, or acts as a link/button, the container tag should be given `aria-hidden="true"` to ensure that screen readers do not present this information to the user as they encounter it.

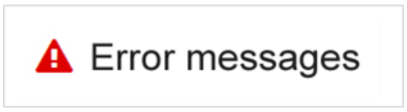
If possible, the tag serving the icon should be a `<span>` because it has no semantic value.

3 cases:

- Decorative/ambient icons.
- Informative icons.
- Link or button icons.

Decorative/ambient icons

When a decorative or ambient icon is included in the HTML code there is no need for an alternative.



Error messages

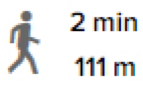
In the following HTML code, the icon is simply a visual embellishment of the text "Error messages" (which is explicit by nature):

```
<h2>
   <span class="icon error" aria-hidden="true"></span>
   Error messages
</h2>
```

Informative icons

When an informative icon is included in the HTML code:

1. Add a tag (e.g. `<span>`) immediately after the tag which contains the icon.
2. Add information to this tag to make the icon explicit.
3. Hide the content of the tag with CSS by placing it off-screen (outside the viewport) using the CSS "`position: absolute;`" property.



In the example of HTML code below, the icon refers to the time and distance to reach a destination **by walking**:

```
<p>
  <span class="icon walking" aria-hidden="true"></span>
  <span class="off-screen">By walking: </span>
  <span>2 min</span>
  <span>111 m</span>
</p>
```

```
.off-screen {
   position: absolute;
   left: -99999rem;
}
```

Links or buttons icons

When an unlabelled icon acting as a link or button is added to the HTML code:

1. Add a `<span>` tag inside the `<a>` or `<button>` tag.
2. Add text to the `<span>` tag to make the link or button explicit.
3. Hide the content of the tag with CSS by placing it off-screen (outside the viewport) using the CSS "`position: absolute;`" property.

In the example of HTML code below, the icon is a link that points to the homepage:

```
<a href="/">
   <span class="icon home" aria-hidden="true"></span>
   <span class="off-screen">Home</span>
</a>
```

```
.off-screen {
   position: absolute;
   left: -99999rem;
}
```

6.4 Tag images that have captions with `<figure role="figure">` and `<figcaption>`

Images that have captions must be tagged with `<figure role="figure">` and `<figcaption>`.

The `<figure role="figure">` tag must:

- Include the image and the legend, and the legend must be tagged with `<figcaption>`.
- Have an `aria-label` attribute that must have the same content as the `<figcaption>` tag.

Note

Ideally, the alternative text for the image (i.e. `alt` attribute for the `<img />` tag) must contain a reference to the adjacent caption.

Example 1

Bolivian desert, picture by Audesou.

```
<figure role="figure" aria-label="Desert in Bolivia, photo by
Audesou.">
   <img src="…" alt="Three lamas in the Bolivian desert." />
   <figcaption>
      Desert in Bolivia, photo by Audesou.
   </figcaption>
</figure>
```

Example 2

Photo 1: shot by Audesou in Bolivia.

```
<figure role="figure" aria-label="Photo 1 : shot by Audesou in
Bolivia.">
   <img src="…" alt="Three lamas in the desert. [photo 1]" />
   <figcaption>
      Photo 1: shot by Audesou in Bolivia.
   </figcaption>
</figure>
```

7. Forms

7.1 Use the `<label>` tag with the for and `id` attributes to label form fields with a visible name

To label fields with a visible name:

1. Use the `<label>` tag for each form name.
2. Add the for attribute to each `<label>` tag and an `id` attribute on each form field.
3. Populate the `for` and `id` attributes of each name/field pair with the same, unique value.

```
<label for="name">Your last name</label>
<input type="text" id="name" name="name" autocomplete="family-
name" />
```

```
<label for="country">Your country</label>
<select id="country" name="country" autocomplete="country-
name">
   <option value="belgium">Belgium</option>
   <option value="france">France</option>
   […]
</select>
```

Note

It is important to provide identical names for fields that have the same function.

For example, if there are several identification forms on the website, do not use "ID" on one, and "Login" on the other.

7.2 Use the `title` attribute to label form fields that don't have a visible label

To label form fields that don't have a visible label:

1. Add `title` to the field.
2. Populate the attribute by specifying the field's function.

```
<input type="search" title="Your search" name="search" />
<input type="submit" value="Search" />
```

```
<select title="Filter news" name="filter">
   <option>By Date</option>
   <option>By Topic</option>
   […]
</select>
```

Warning

The `placeholder` attribute cannot be used as a label because it disappears during text entry.

It should only be used for secondary text entry aids that are not essential for understanding the field.

```
<input type="text" title="Your search" placeholder="Enter
keywords" />
```

7.3 Use the `autocomplete` attribute to facilitate auto-fill

To facilitate auto-fill on fields referring to personal information:

1. Add an `autocomplete` attribute to the field.
2. Populate the `autocomplete` attribute according to the type of information expected.

```
<label for="first-name">Your first_name</label>
<input type="text" id="first-name" name="first-name"
autocomplete="given-name" />
```

Note

The possible values for the `autocomplete` attribute are standardized.

7.4 Add help text directly to the `<label>` tags

Help text must be included directly in the `<label>` tags.

This particularly applies to:

- Indications such as "Required field".
- Instructions on how to format the input such as MM/DD/YYYY.
- Information about the type and maximum size of files than can be sent.
- Etc.

```
<label for="date">
   Date of Birth
   <span>(MM/DD/YYYY)</span>
</label>
<input type="date" id="date" name="date" autocomplete="bday"
/>
```

```
<label for="number">
    Your client number
    <input type="text" id="number" name="number" aria-
required="true" />
    <span>i.e., BT-VZ</span>
</label>
```

Note

In some cases, because of a specific design layout for example, the help text cannot be included in the `<label>`.

In these cases:

1. Add the `id` attribute to the tag containing the help text.

2. Populate the `id` with a unique value.

3. Add the `aria-describedby` attribute to the corresponding field.

4. Populate `aria-describedby` with the `id` value included in the help text.

```
<label for="document">
    Add a document to your case
 </label>
 <input type="file" id="document" name="document" aria
 describedby="formats" />
 <h2> Documents currently in your case</h2>
 <ul>
    <li>CV</li>
    <li>Cover Letter</li>
 </ul>
 <p id="formats">Accepted formats: pdf or doc.</p>
```

Note that the `aria-describedby` attribute can reference several `id` values.

Warning

The `placeholder` attribute must not be used for entry aids that are essential for understanding the type of data to be entered.

It must be used only for secondary entry aids.

```
<input type="search" title="Your search" placeholder="Enter
keywords" />
```

7.5 Add `required` or `aria-required="true"` to mandatory fields

Required fields must have a `required` or `aria-required="true"` attribute.

Note

In addition to this attribute, a distinctive sign must be included in the `<label>` tag.

```
<label for="email">Your email *</label>
<input type="email" id="email" name="email"
autocomplete="email" required />
```

```
<input type="checkbox" id="conditions" aria-required="true" />
<label for="conditions">I accept the conditions of sale
(required)</label>
```

7.6 Add error messages and correction suggestions directly to the `<label>` tag

When error messages and suggested corrections are placed next to the relevant form fields, they need to be added directly into the <label> tags.

Tip

In addition to the error message, good practice is to add `aria-invalid="true"` to the field.

```
<label for="name">
   Your name *
   <span>Please fill in your name</span>
</label>
<input type="text" id="name" name="name" autocomplete="family-
name"  aria-required="true" aria-invalid="true" />
```

```
<label for="email">
   Your email *
   <input type="email" id="email" name="email"
autocomplete="email" aria-required="true" aria-invalid="true"
/>
   <span>Please provide a valid email format
(example@domain.com)</span>
</label>
```

Note

In some cases, because of a specific design layout for example, the error message cannot be directly included in the `<label>` tag.

In that case:

1. Add an `id` attribute to the element that contains the error message.

2. Give this `id` a unique value.

3. Add `aria-describedby` to the corresponding input field.

4. Add the `id` value of the error message to this `aria-describedby` attribute.

```
<label for="document">
   Add a document to your file
</label>
<input type="file" id="document" name="document" aria-
invalid="true" aria
describedby="formats error" />
<h2>Documents currently in your file</h2>
<ul>
   <li>CV</li>
   <li>Cover letter</li>
</ul>
<p id="error">File format incorrect.</p>
<p id="formats">Valid formats: pdf or doc.</p>
```

Note that `aria-describedby` can reference several `id` values.

7.7 Group and label related fields with `<fieldset>` and `<legend>`

When a form has multiple fields and some of them have the same label, use `<fieldset>` and `<legend>` to group them.

```
<fieldset>
   <legend>Participant 1</legend>
   <label for="firstName-1">First Name</label>
   <input type="text" id="firstName-1" name="firstName-1" />
   <label for="lastName-1">Last Name</label>
   <input type="text" id="lastName-1" name="lastName-1" />
   […]
</fieldset>
<fieldset>
   <legend>Participant 2</legend>
   <label for="firstName-2">First Name</label>
   <input type="text" id="firstName-2" name="firstName-2" />
   <label for="lastName-2">Last Name</label>
   <input type="text" id="lastName-2" name="lastName-2" />
   […]
</fieldset>
```

Warning

The `<fieldset>` and `<legend>` tags must systematically be used when several groups of fields have identical labels.

For example:

- A series of different questions on the same page which can all be answered “yes” or “no”.
- A list of participants at an event, featuring the First Names and Last Names of each participant.

If the form is too long, but none of the field groups have the same label, use the tags `<h1>` to `<h6>` to sort the fields groups.

Note

The use of `<fieldset>` and `<legend>` tags is particularly necessary when integrating radio button or checkbox lists.

For example:

```
<fieldset>
  <legend>Sports played</legend>
  <ul>
```

```
    <li>
      <input type="checkbox" id="basketball" />
      <label for="basketball">Basketball</label></li>
    <li>
      <input type="checkbox" id="tennis" />
      <label for="tennis">Tennis</label></li>
    […]
  </ul>
</fieldset>
```

8. Lists

8.1 Tag unordered lists with `<ul>` and `<li>`

Use the `<ul>` and `<li>` tags to mark up lists of items which can appear in any order (menus, tabs, sharing buttons, site map, etc.).

If required, make sure that lists are properly nested:

```
<ul>
   <li>First item of first level</li>
   <li>
      Second item of first level
      <ul>
         <li>First item of second level</li>
         <li>Second item of second level</li>
      </ul>
   </li>
   <li>Third item of first level</li>
</ul>
```

What are the advantages?

Structuring lists is essential for **people using a screen reader (blind or visually impaired)**. When they come across a structured list, they are able to:

- Navigate from list to list within a page.
- Know from the start the number of elements in the list.
- Navigate more easily through the list:
  - Go directly to the end of the list if they are not interested in the content.
  - Easily return to the top of the list.

8.2 Tag ordered lists with `<ol>` and `<li>`

Use the `<ol>` and `<li>` tags to mark up lists of items that must appear in a specific order (list of steps in a procedure, ranking, etc.). In other words, when the information would not be understood if the items were presented in a different order.

If required, make sure that lists are properly nested:

```
<ol>
   <li>First item of first level</li>
   <li>
      Second item of first level
      <ul>
```

```
         <li>First item of second level</li>
         <li>Second item of second level</li>
      </ul>
   </li>
   <li>Third item of first level</li>
</ol>
```

What are the advantages?

Structuring lists is essential for **people using a screen reader (blind or visually impaired)**. When they come across a structured list, they are able to:

- Navigate from list to list within a page.
- Know from the start the number of elements in the list.
- Navigate more easily through the list:
  - Go directly to the end of the list if they are not interested in the content.
  - Easily return to the top of the list.

8.3 Mark up description lists with `<dl>`, `<dt>` and `<dd>`

Use `<dl>`, `<dt>` and `<dd>` tags to mark up description lists.

A description list is a series of key/value pairs that might be found, for example, in a product description, or in a glossary of terms.

Note

One keyword (`<dt>` tag) might have several values (`<dd>` tags).

Example with the description of an event:

```
<h2>Web Accessibility Conference</h2>
<dl>
   <dt>Location</dt>
   <dd>Paris</dd>
   <dt>Dates</dt>
   <dd>Saturday, September 7</dd>
   <dd>Wednesday, October 14</dd>
   <dt>Time</dt>
   <dd>Starts at 10 a.m.</dd>
</dl>
```

Example with a glossary:

```
<dl>
   <dt><dfn>ARIA</dfn></dt>
   <dd>Accessible Rich Internet Application</dd>
   […]
   <dt><dfn>WCAG</dfn></dt>
   <dd>Web Content Accessibility Guidelines</dd>
   […]
</dl>
```

What are the advantages?

Structuring lists is essential for **people using a screen reader (blind or visually impaired)**. When they come across a structured list, they are able to:

- Navigate from list to list within a page.
- Know from the start the number of elements in the list.
- Navigate more easily through the list:
    - Go directly to the end of the list if they are not interested in the content.
    - Easily return to the top of the list.

9. Tables

9.1 Use `<caption>` to tag data table captions

When a data table is introduced by a caption, it must be tagged using the `<caption>` tag.

Note

The caption must be precise and concise.

```
<table>
   <caption>Average monthly temperatures of the 3 largest
cities in Canada.</caption>
    […]
</table>
```

9.2 Mark up the row and column header cells with `<th>`

In all data tables, mark up each column and row header cell with the `<th>` tag.

When a cell is essential to understand the data provided in the table, it must be marked up with `<th>`.

```
<table>
   <caption>Average monthly temperatures of the 3 largest
cities in Canada.</caption>
   <tr>
      <td>&nbsp;</td>
      <th>Toronto</th>
      <th>Montreal</th>
      <th>Vancouver</th>
   </tr>
   <tr>
      <th>June</th>
      <td>22°C</td>
      <td>28°C</td>
      <td>26°C</td>
   </tr>
   <tr>
      <th>July</th>
      <td>24°C</td>
      <td>30°C</td>
      <td>28°C</td>
   </tr>
</table>
```

Average monthly temperatures of the 3 largest cities in Canada.

	Toronto	**Montreal**	**Vancouver**
June	22°C	28°C	26°C
July	24°C	30°C	28°C

9.3 In simple data tables use the `scope` attribute to associate data cells with their corresponding headers

A data table is simple when each header cell applies to all the cells in the row or column.

To associate headers with their corresponding data in this type of table, add the `scope` attribute to the `<th>` tags. The value of the attribute depends on whether the header cell concerns:

- An entire column: `scope="col"`.
- An entire row: `scope="row"`.

```
<table>
   <caption>Average monthly temperatures of the 3 largest
cities in Canada.</caption>
   <tr>
      <td>&nbsp;</td>
      <th scope="col">Toronto</th>
      <th scope="col">Montreal</th>
      <th scope="col">Vancouver</th>
   </tr>
   <tr>
      <th scope="row">June</th>
      <td>22°C</td>
      <td>28°C</td>
      <td>26°C</td>
   </tr>
   <tr>
      <th scope="row">July</th>
      <td>24°C</td>
      <td>30°C</td>
      <td>28°C</td>
   </tr>
</table>
```

Average monthly temperatures of the 3 largest cities in France.

	Paris	**Marseille**	**Lyon**
June	22°C	28°C	26°C
July	24°C	30°C	28°C

9.4 Add `role="presentation"` to each layout `<table>`

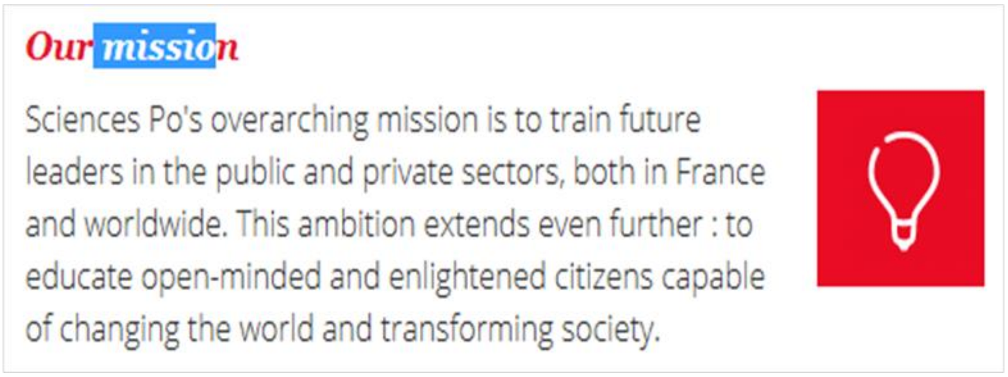
Mark up layout tables by including `role="presentation"` in the `<table>` tag.

```
<table role="presentation">[…]</table>
```

10. Using CSS

10.1 Use CSS to lay out text

Texts must not be integrated as images but as texts in HTML format stylized with CSS.



Our mission

Sciences Po's overarching mission is to train future leaders in the public and private sectors, both in France and worldwide. This ambition extends even further : to educate open-minded and enlightened citizens capable of changing the world and transforming society.

Example of a specific font added in text format

10.2 Ensure that content can be read when images are turned off

When images are turned off, the page content must remain visible and readable. No information must be lost and there must be sufficient contrast between the foreground and background.

In other words, all the content must be visible and readable:

- Even when CSS images are turned off for the page.
- Even when HTML images are replaced with their `alt` attributes.

Note

Whenever text is placed over a background image, make sure to provide an alternative background color that will ensure text readability in the absence of the background image.

```
background: black url(../images/dark-background.png) repeat-x;
```

This replacement color can be inherited from a parent.

Warning

Special attention must be paid to the readability of alternative text for images embedded in the HTML code when images are turned off.

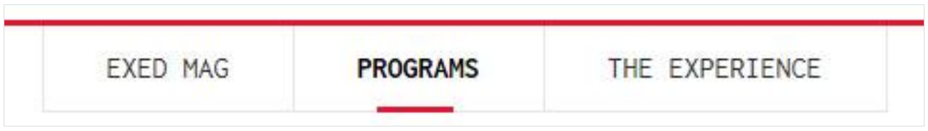
10.3 Ensure that content is understandable when CSS is disabled

Ensure that the information remains available and understandable even when CSS is disabled, especially when colors, sizes, shapes or positions are information carriers.

Also, make sure not to generate informative content using only CSS.

Examples

Position of the current selection in the menu (Example 1)

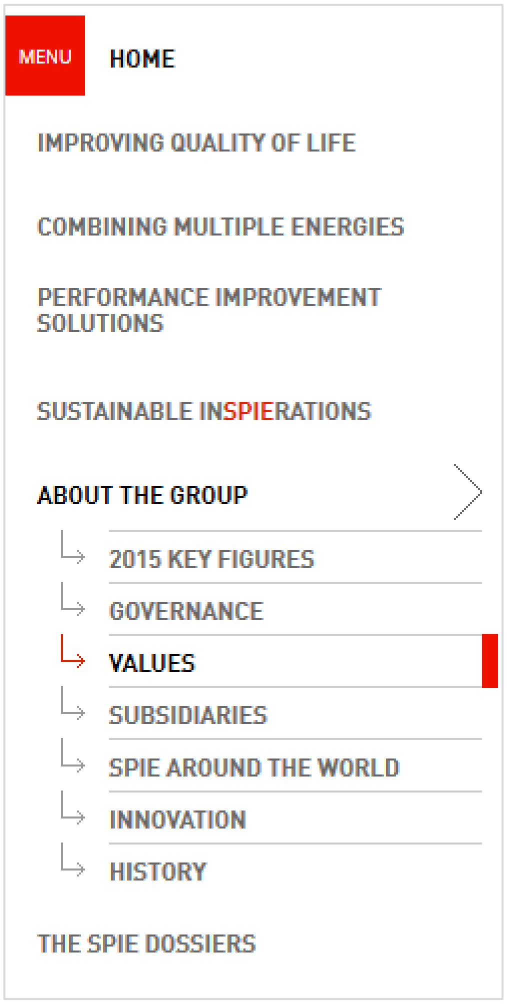


In this example, "Programs" has bold text and a red border indicating it is currently selected.

In the solution presented below, a `<strong>` tag is added to ensure that the information is not lost without CSS.

```
<a href="…" aria-current="true">
   <strong>Programs</strong>
</a>
```

Position of the current selection in the menu (Example 2)



In this example, a different colored arrow and an additional border indicate that the link "Values" is the current page.

In this case, an efficient way of conveying that it is the current page without using CSS is to remove the surrounding `<a>` tag.

```
<ul>
   <li><a href="…">2015 key figures</a></li>
   <li><a href="…">Governance</a></li>
   <li aria-current="page">Values</li>
   <li><a href="…">Subsidiaries</a></li>
   <li><a href="…">SPIE around the world</a></li>
   <li><a href="…">Innovation</a></li>
   <li><a href="…">History</a></li>
</ul>
```

CSS Generated Content (Example 3)

When informative content is necessary to understand the content, it must not be generated by CSS.

❌
```
a[href*=".pdf"]::after {
   content: ' (PDF)';
}
```

In this example, information about the PDF format of the downloadable files is added with CSS. This is incorrect.

❌
```
label.required::after {
   content: ' *';
}

input[aria-required=true]::before {
   content: '* ';
}
```

In these examples, the asterisk flagging up the mandatory fields of a form is added using CSS, which is also incorrect.

11. Zoom and font sizes

11.1 Use only relative font sizes (`rem`, `em`, `%`, etc.)

To set the font size, use only relative units in CSS font-size such as `rem`, `em`, `%` or keywords (`x-small`, `small`, etc.).

Do not use absolute units such as `pt`, `cm`, etc.

Warning

Although the pixel (px) can be used to define font size, it can prevent font size increase in some cases.

We strongly advise against using it.

Note

This recommendation does not apply to CSS print (media type `print`).

11.2 Ensure content readability even when font size is doubled

Guarantee content readability even when font is twice the default size.

Across the entire font size increase range, content and functionalities must not overlap or disappear.

To ensure this rule is respected:

- Use only relative units (`rem`, `em`, `%`, etc.) to manage spaces (CSS properties `margin` and padding).
- Do not use units (`px`, `pt`, `em`, `%`, etc.) with the CSS `line-height` property.
- Do not define a fixed height (CSS `height` property) on elements likely to contain text, particularly form fields.
- Be careful when using absolute positioning (CSS declaration: `position: absolute`;). Although this type of positioning is compatible with accessibility, it can cause content overlap in some cases.

12. Keyboard Navigation

12.1 Ensure the proper functioning of the interface using either the mouse or keyboard

All interactions must be possible using either the mouse or the keyboard.

In other words, whenever the mouse can be used to interact with a component, the same must also be true for the keyboard.

This is the case for example when displaying/hiding:

- A sub-menu.
- A modal window.
- A customized tooltip.

Warning

Whether keyboard navigation is forwards or backwards, the page must not contain any keyboard traps.

I.e. the keyboard focus must never:

- Remain blocked on an element without any way out.
- Loop in zone on the page without any way out.

Note

Make sure that mouse and keyboard interactions also work on touch screens (and vice versa).

12.2 Ensure that the tab order follows the logical reading order

The tab order must logically follow the visual order of the page.

In other words, if an interactive element follows another interactive element in the visual presentation of the page, the focus must also move from the first to the second element in the same order.

Warning

Do not use positive values (value above 0) on the `tabindex` attribute because it can disrupt the natural tab order of the page.

Note

In some rich interface components such as tabbed interfaces, the arrow keys rather than the Tab key are used to navigate from one interactive element to another.

12.3 Ensure that keyboard focus is visible

In order to ensure that users navigating using the keyboard know where they are on the page, each interactive element (links, buttons, form fields, etc.) must stand out visually when it receives focus.

Note

Ensure that this focus indicator has sufficient contrast.

Tip

Good accessibility practice is to systematically back up each `:hover` rule with a `:focus` rule in CSS.

For example:

```
main a:hover,
main a:focus {
   text-decoration: none;
}
```

12.4 Introduce “skip links”

A "Skip to content" must be placed on each page to help keyboard users navigate the page.

This should be the first interactive element in the HTML code.

This internal link will take the user directly to the main content.

```
<body>
<a class="skiplink" href="#content">Skip to content</a>
[…]
<main role="main">
     <a id="content" tabindex="-1"></a>
     […]
</main>
[…]
</body>
```

Note

The link may be hidden by default. However, it must become visible when it receives keyboard focus.

Therefore, the "skip to content" link should never be hidden using CSS properties `display: none;` and/or `visibility: hidden;` which would make it completely unusable with keyboard navigation.

A better approach would be to use the code below:

```
a.skiplink {
   display: inline-block;
   color: #555;
   background: #fff;
   padding: .5em;
   position: absolute;
   left: -99999rem;
   z-index: 100;
}

a.skiplink:focus {
   left: 0;
}
```

Tip

In some situations, it takes a lot of tabbing to reach the main and secondary menus and/or search engine from the top of the page.

In these cases, add "Skip to" links, as in the following example:

```
<ul id="skiplink">
   <li>
      <a href="#content">Skip to content</a>
   </li>
   <li>
      <a href="#nav">Skip to navigation</a>
   </li>
   <li>
      <a href="#search">Skip to search</a>
   </li>
</ul>
```

13. Iframes

13.1 Populate the title attribute on each `<iframe`

Whenever an `<iframe>` tag is included in the page, it is also necessary to add a `title` attribute.

The attribute value must introduce the content of the `<iframe>` tag clearly and concisely.

For example, in the case of an `<iframe>` loading a video player on the page:

```
<iframe src="https://www.youtube.com/embed/y525BrxyvhA"
title="Vidéo YouTube : L'accessibilité numérique à toutes les
étapes d'un projet">
   […]
</iframe>
```

14. Additional Guidelines

Some guidelines found in accessibility reference documents were not included in this manual as they are considered as rarely applicable.

It is however necessary to respect these additional rules to ensure compliance with RGAA 4 and WCAG 2.1.

They are listed below:

- Make sure the information is understandable, even without color
- Ensure content readability even when text spacing properties are customized
- Incorporate the summary of each complex data table in the `<caption>` tag
- Provide alternative text for each informative `<canvas>`, `<embed>` and `<object>` tag
- Provide alternative text for each `<canvas>`, `<embed>` and `<object>` tag used as a link or button.
- Hide each decorative or ambient `<canvas>`, `<embed>` and `<object>` tag on assistive technologies
- Do not initiate an action while the trigger is in the "pressed" state
- Anticipate the interface behavior
- Group related options in `<select>` boxes with `<optgroup>`
- Add the title attribute to describe each `<frame>`
- Use the `dir` attribute to indicate a change in reading direction.

Tag block quotations using `<blockquote>`

Whenever, a block quotation is to be included in the HTML page, enclose it in a `<blockquote>` tag.

Note

Any isolated quote that is not directly incorporated in the surrounding text flow, is considered as a block quotation.

Tag online quotations using `<q>`

Whenever a quotation is integrated in a text, enclose it in the `<q>` tag.

Write the HTML code in a logical reading order

The writing order of the tags in the flow of the HTML code must follow the reading order of the page.

In other words, when one tag precedes another in the visual flow of the page, it must also immediately precede that element in the HTML code.

Tip

To test this rule, turn off the CSS and check that the resulting page order matches the page when CSS is active.

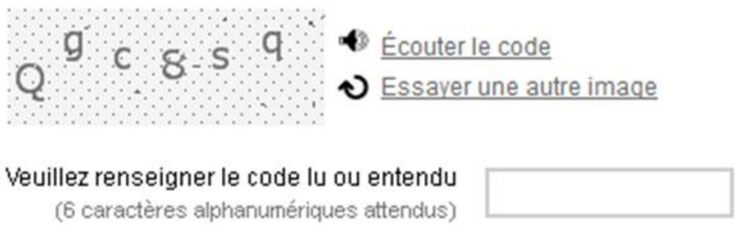
Warning

If some content is hidden by default, make sure that it is properly positioned in the HTML flow when styles are disabled.

Indicate in the alternative text for visual CAPTCHA where to find the non-visual version

Whenever a visual CAPTCHA is used, indicate in the alternative text the information needed to:

- Identify it as a security system.
- Find the non-visual version of the CAPTCHA.



Veuillez renseigner le code lu ou entendu
(6 caractères alphanumériques attendus)

It is often an audio version, so in this case opt for:

```
<img src="captcha.png" alt="Spam protection system: If you
can't use it, use the audio version available below." />
```

Note

To correctly enter the alternative text in compliance with the CAPTCHA integration technique, refer to sheets on the topic Images and icons.

Do not use tags or attributes specific to data tables in layout table

A layout table must not have any tags or attributes specific to data tables.

This means that:

- The tags `<caption>`, `<colgroup>`, `<tfoot>`, `<th>` and `<thead>` must not be used in layout tables.
- The attributes `axis`, `headers` and `scope` must not be used in layout tables.

Plan alternative text for each type of multimedia content (`<video>`, `<audio>`, `<object>`, `<embed>`, etc.)

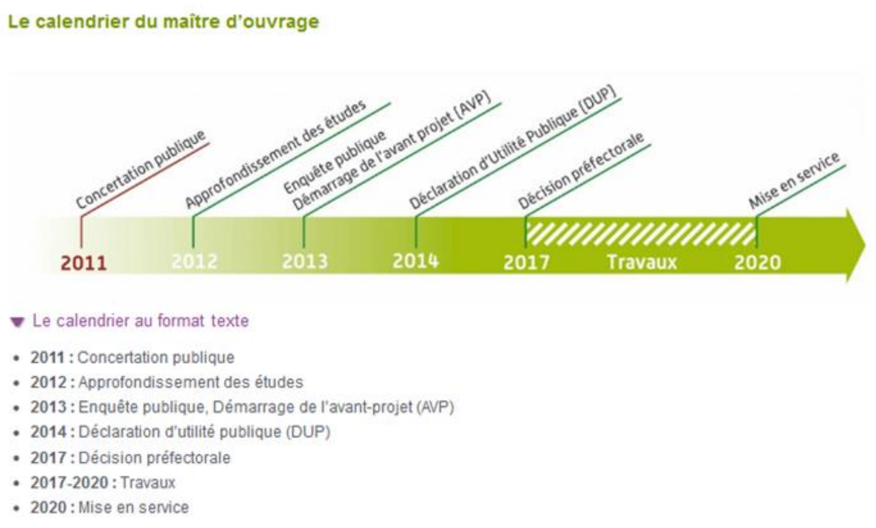
Plan alternative text for each type of multimedia content embedded in the page using the tags `<video>`, `<audio>`, `<object>`, `<embed>`, for example.

This alternative text will provide the same level of information to people who cannot access the multimedia content (no plugin, old browser, etc.).

It can take the form of HTML content for example.

Tip

When a video is displayed on a page, its transcription can be considered as an alternative.



Le calendrier du maître d'ouvrage

▼ Le calendrier au format texte

- **2011 :** Concertation publique
- **2012 :** Approfondissement des études
- **2013 :** Enquête publique, Démarrage de l'avant-projet (AVP)
- **2014 :** Déclaration d'utilité publique (DUP)
- **2017 :** Décision préfectorale
- **2017-2020 :** Travaux
- **2020 :** Mise en service

In this example, the calendar is a video carrying information, and a link "The calendar in text format" leads to alternative text in HTML format.

Populate the `alt` attribute of each mapped image and its `<area />` tags

When a mapped image is integrated in the HTML code:

- An attribute must be added to the `<img />` tag and to each `<area />` tag.
- The alt attribute of the `<img />` tag must announce the function of the mapped image.
- The alt attribute of each `<area />` tag must express the destination of the link.

For example, in the case of a map of France where each department is clickable and leads to an information sheet about the department:

```
<img src="…" alt="Map of France" usemap="#map-France" />
<map name="map-France">
  <area shape="poly" coords="…" href="…" alt="Ain" />
  <area shape="poly" coords="…" href="…" alt="Aisne" />
  <area shape="poly" coords="…" href="…" alt="Allier" />
  […]
</map>
```

In complex tables, use the `headers` and `id` attributes to associate header cells with their corresponding data cells

A data table is complex when the header cells do not apply to all the cells in the row or column.

To associate headers with their corresponding data cells in this type of table, add the `id` attribute to all `<th>` cells and the `headers` attribute to all the `<td>` cells.

The next step is to populate the `headers` attributes with the values found in the corresponding header cells. If several headers are associated with a data cell, they are separated by a space (space delimited) in the corresponding `headers` attribute.

```
<table>
   <caption>Sales revenue comparison of Davis and Davison
companies in the UK and in the rest of the world.</caption>
   <tr>
       <th id="header-1">In millions of pounds sterling</th>
       <th id="header-2">In the UK</th>
       <th id="header-3">Rest of the World</th>
   </tr>
   <tr>
       <th id="header-4">Davis</th>
       <td headers="header-4 header-2 header-1">
           50.7
       </td>
```

```
        <td headers="header-4 header-3 header-1">
            139.3
        </td>
    </tr>
    <tr>
        <th id="header-5">Davison</th>
        <td headers="header-5 header-2 header-1">
            27.1
        </td>
        <td headers="header-5 header-3 header-1">
            476.0
        </td>
    </tr>
</table>
```

Comparison of sales revenue between the companies Dupond and Dupont in France and the rest of the world.

In millions of euros	In France	Rest of the World
Dupond	50.7	139.3
Dupont	27.1	476.0

Warning

The `headers` attribute must not be used in combination with the `scope` attribute.

Note

Good practice is to organize the `id` values of the header cells according to the logical visual order of the table.

A screen reader (using voice or braille) will announce the headers in this order.

Ensure that the HTML flow is consistent from one page to the next

The order of the main content blocks in the HTML flow must remain consistent from one page to the next.

For example, if the secondary menu is placed after the main content, it is important to keep this order on all pages of the website.

Make sure layout tables follow the right reading order

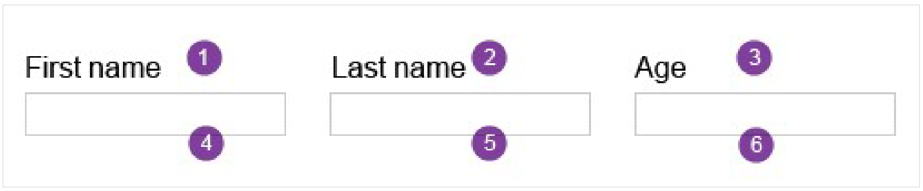
When a screen reader (voice or braille) translates a layout table, the content is read in a linear way i.e. cell by cell, from left to right, row after row.

So the reading order should remain consistent on each layout table.

For instance, if the source code below is used, the reading order will be:

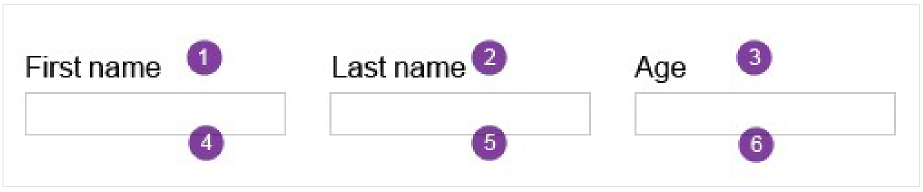
1. First name.
2. Last name.
3. Age.
4. “First name” field.
5. “Last name” field.
6. “Age” field.

```
<table role="presentation">
 <tr>
  <td><label for="first-name">First name</label></td>
  <td><label for="last-name">Last name</label></td>
  <td><label for="age">Age</label></td>
 </tr>
 <tr>
  <td><input type="text" name="first-name" id="first-name"
/></td>
  <td><input type="text" name="last-name" id="last-name"
/></td>
  <td><input type="text" name="age" id="age" /></td>
 </tr>
</table>
```



To follow a coherent reading order while keeping the layout table, try this instead:

```
<table role="presentation">
 <tr>
  <td>
   <label for="first-name">First name</label>
   <input type="text" name="first-name" id="first-name" />
  </td>
  <td>
   <label for="last-name">Last name</label>
   <input type="text" name="last-name" id="last-name" />
  </td>
  <td>
   <label for="age">Age</label>
   <input type="text" name="age" id="age" />
  </td>
 </tr>
</table>
```



Warning

To ensure optimal screen reader compatibility and guarantee that the right reading order is followed, it is strongly recommended to limit the use of nested layout tables

Additional recommendations

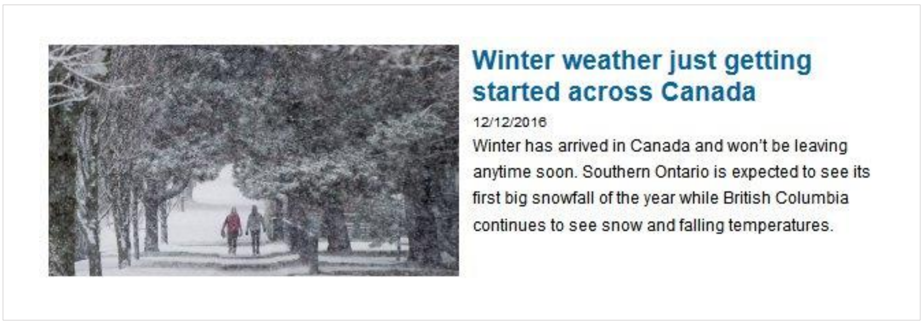
In addition to the mandatory rules detailed in the previous chapters of this manual, taking into account the additional recommendations below is beneficial for people with disabilities.

We therefore strongly recommend that you apply them.

Tag abbreviations with `<abbr>`

Combine adjacent links pointing to the same page

When several links pointing to the same page are grouped together, they should be combined in a single tag `<a>`.



Winter weather just getting started across Canada

12/12/2016

Winter has arrived in Canada and won't be leaving anytime soon. Southern Ontario is expected to see its first big snowfall of the year while British Columbia continues to see snow and falling temperatures.

Here is an example of coding for an image, a title, a date and an introduction serving as a link to the details of a news item **that we do not recommend**:

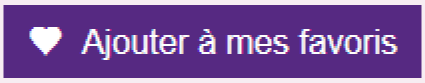
```
<a href="…">
   <img src="people-in-the-snow.jpg" alt=" Winter weather just
getting started across Canada" />
</a>
<h2><a href="…">Winter weather just getting started across
Canada</a></h2>
<p class="date">
   <a href="…">12/12/2016</a>
</p>
<p class="intro">
   <a href="…"> Winter has arrived in Canada and won’t be
leaving anytime soon. Southern Ontario is expected to see its
first big snowfall of the year while British Columbia
continues to see snow and falling temperatures.</a>
</p>
```

And here is a more accessible coding:

✓
```
<a href="…">
   <img src="people-in-the-snow.jpg" alt="" />
   <h2>Winter weather just getting started across Canada</h2>
   <p class="date">12/12/2016</p>
   <p class="intro">Winter has arrived in Canada and won’t be
leaving anytime soon. Southern Ontario is expected to see its
first big snowfall of the year while British Columbia
continues to see snow and falling temperatures.</p>
</a>
```

Note

This recommendation also applies to `<button>` tags.



Here is an example of coding of a grouped icon and text serving as an Add bookmark button **that we do not recommend**:

```
<button>
   <img src="favorites.png" alt=" Add to favorites" />
</button>
<button>Add to favorites</button>
```

And here is a **more accessible** coding:

```
<button>
   <img src=" favorites.png" alt="" />
   Add to favorites
</button>
```

In forms with multiple steps, identify the current step using `aria-current="step"`

In the navigation menu of a form with multiple steps, the current step should be identified by means of `aria-current="step"`.

```
<nav aria-label="Steps of your order">
   <ol>
      <li><a href="…">Personal information</a></li>
      <li aria-current="step"><em>Payment</em></li>
      <li>Preview</li>
   </ol>
</nav>
```

Note

When forms have multiple steps, it should also be possible for the user to return to a previous step by means of a link, for example.

Identify the current position in navigation systems with `aria-current`

In all navigation systems (menus, breadcrumb trails, pagination systems, etc.), the current position is to be identified by `aria-current="page"`.

Likewise, any parent sections and sub-sections are to be identified by `aria-current="true"`, only in menus (and not in breadcrumb trails) by.

Example in a menu

In the example below of code for a navigation menu, the parent section “Upcoming events” is identified by `aria-current="true"` included in the `<a>` tag and the current page “Education” is identified by `aria-current="page"` included in the `<li>` tag.

```
<nav role="navigation" aria-label="Main menu">
   <ul>
      <li><a href="…">Home</a></li>
      <li><a href="…" aria-current="true">Upcoming events</a>
         <ul>
            <li><a href="…">Culture</a></li>
            <li aria-current="page">Education</li>
            <li><a href="…">Sports</a></li>
         </ul>
      </li>
      <li><a href="…">News</a></li>
      <li><a href="…">Contact</a></li>
   </ul>
</nav>
```

Example in a pagination system

In the code example below for a pagination system, the current page "2" is identified by `aria-current="page"` included in the `<li>` tag.

```
<nav aria-label="Pagination">
   <ul>
      […]
      <li><a href="…" aria-label="Page 1">1</a></li>
      <li aria-current="page">2</li>
      <li><a href="…" aria-label="Page 3">3</a></li>
      […]
   </ul>
</nav>
```

Example in a breadcrumb trail

In the example below of code for a breadcrumb trail, the current page "Editorial manual" is identified by `aria-current="page"` included in the `<li>` tag.

```
<nav aria-labelledby="intro-pagination">
   <p id="intro-pagination">You are here:</p>
   <ul>
      <li><a href="…">Home</a></li>
      <li><a href="…">AcceDe Web guidelines</a></li>
      <li aria-current="page">Editorial manual</li>
   </ul>
</nav>
```

Add file format and size to all Download links and buttons

Whenever a link or button points to a download file, the following information should be added to the label:

- Document name.
- Document format.
- Document size.

```
<a href="Toronto bus service.pdf">
Download the map of Toronto bus routes (PDF - 2 MB)
</a>
```

Note

If an icon is used to indicate the file format, refer to "Images and Icons", to implement it in an accessible way.

Tip

In situations where the size of the downloadable file cannot be known in advance (i.e. generation on the fly), good accessibility practice is to give an approximate or maximum size.

For example:

```
<a href="invoice.pdf">
Download your March invoice (PDF - Approx. 1.5MB)
</a>
```

Update the page `<title>` when an error or confirmation message is displayed

Whenever a form returns an error or confirmation message, it is advisable to update the `<title>`.

In the case of a confirmation message:

```
<title>Confirmation - Contact | [site name]</title>
```

If there is an error:

```
<title>Error - Contact | [site name]</title>
```

Tip

It is also recommended to have this information appear first in the `<title>` tag.

Note

In some situations, the title does not need to be updated because the title displayed after submission is already obvious.

For example

- A login form which goes directly to a profile page.
- A button which says “Next step” as part of a multi page form.
- A contact page that sends a preview page.

Do not disable the zoom with the `user-scalable` property

It is important to be able to zoom in or out of a page. For this purpose, the following declarations must not be used:

```
<meta content="width=device-width;initial-scale=1.0; maximum-
scale=1.0; user-scalable=1;" name="viewport" />
```

```
<meta name="viewport" content="user-scalable=no" />
```

Do not use CSS to display images that carry information

CSS should not be used to display images that carry information.

In other words, images with information should be hard-coded in the HTML (i.e. via the `<img />` or `svg` tag).

Note

Image sprites (images loaded in the background via CSS) are not appropriate for images that carry information.

Tip

Images are considered as information carriers if some of the page content or functions were missing without them. These images include:

- Logos.
- Text images.
- Images used as links or buttons.
- Etc.

Warning

In particular, we strongly advise against the technique of placing text off-screen and replacing it with a background image.

When the images are not uploaded, the information may be lost.

❌
```
<a href="/" id="logo">[My logo text]</a>
```

❌
```
#logo {
    display: block;
    width: 300px;
    height: 100px;
    text-indent: -99999rem;
    background: url('images/logo.png');
}
```

Pure HTML should be used instead.

Logically organize the items of dropdown lists

When dropdown lists are used, the options listed should be organized logically.

The logical order depends on the context. Examples include:

- Alphabetical order (e.g. a list of languages).
- Numbered order (e.g. a list of French departments).
- Practical order ("France" first in form to sign up for a French government agency).

Tip

Another good accessibility practice is to avoid prefixing the content of the `<option>` tag with decorative characters (dashes, stars, spaces, etc.).

This will give the user direct access to a desired value or group of values by simply pressing a key on the keyboard (pressing the "S" key to reach the country "Spain", for example).

✗

```
<select>
    <option>--Belgium</option>
    <option>--France</option>
    <option>--Germany</option>
    <option>--Italy</option>
    <option>--Spain</option>
 </select>
```

Should be replaced by:

✓

```
<select>
    <option>Belgium</option>
    <option>France</option>
    <option>Germany</option>
    <option>Italy</option>
    <option>Spain</option>
</select>
```

Send a notification when a new window is opened

When a link or button opens a new window in the browser, it is advisable to add information such as “new tab”.

Either in the `aria-label` attribute:

```
<a href="gtc.html" target="_blank" aria-label="General Terms
and Conditions (new tab)">
   General Terms and Conditions
</a>
```

Or in the alternative text of an icon/image as follows:

- Icon Fonts.
- Of an svg (vector graphic).
- Of a tag <img />.

Note

Use of the `title` attribute also complies with accessibility requirements:

```
<a href="gtc.html" target="_blank" title="General Terms and
Conditions (new tab)">
   General Terms and Conditions
</a>
```

Use only relative sizes (`rem`, `em`, `%`, etc.) for media query breakpoints

To define media query breakpoints, it is advisable to use relative units only (such as `rem`, `em` or `%`) for the CSS `@media` rule

```
@media (max-width: 25rem) {
  nav li.tile {
    width: 33%;
  }
}
```

rather than absolute units such as `px`, `pt`, `cm`, etc.

Note

This recommendation also applies for media queries directly integrated in the `<link />` tag:

```
<link rel="stylesheet" href="/css/extra-small.css" media="all
and (max-width: 25rem)" />
```

Make sure that hidden content is translated

For websites in multiple languages or when using scripts developed in a foreign language, make sure to translate into the website's current language all the hidden content in:

- `alt` attributes of images.
- `aria-label` attributes.
- `title` attributes.
- Tags for content not visible on screen.
  (via a CSS class ".`sr-only`", for example)