ATALAN
Accessibilité numérique
Sensibilisation au handicap
www.atalan.fr

AcceDe Web
La démarche accessibilité
www.accede.info

Notice d'accessibilité

HTML, CSS et JavaScript

Date	Version	État / commentaires
10 sept. 2015	2.0	Cette version prend en compte WCAG 2.0 et RGAA 3.0.

En partenariat avec :
Air Liquide – Atos – BNP Paribas – Capgemini – EDF – Generali – SFR – SNCF – Société Générale – SPIE

Et le soutien de :
AbilityNet – Agence Entreprises & Handicap – AnySurfer (*Belgique*) – Association des Paralysés de France (APF) – Association Valentin Haüy (AVH) – CIGREF – Fondation Design For All (*Espagne*) – ESSEC – Handirect – Hanploi – Sciences Po – Télécom ParisTech

Sommaire

Introduction

Contexte et objectifs

Cette notice liste les règles à respecter pour que l'intégration HTML, CSS et JavaScript soit accessible.

Cette notice s'inscrit dans un lot de quatre notices téléchargeables sur le site www.accede-web.com :

- Notice d'accessibilité pour la conception fonctionnelle et graphique.
- Notice d'accessibilité HTML, CSS et JavaScript (présente notice).
- Notice d'accessibilité pour les composants d'interface riche.
- Notice d'accessibilité éditoriale (modèle).

À qui s'adresse ce document ? Comment l'utiliser ?

Ce document doit être transmis aux intervenants et/ou prestataires réalisant les spécifications techniques, l'intégration HTML/CSS et JavaScript. Il vient en complément aux spécifications techniques d'un projet. Les recommandations peuvent être complétées ou retirées selon les contextes d'utilisation, ce travail peut être notamment réalisé par la maîtrise d'ouvrage.

Les recommandations doivent être prises en compte en phase d'intégration HTML/CSS et, pour certaines d'entre elles, lors de la dynamisation des pages lorsque, par exemples, des gabarits sont affectés à un outil de gestion de contenu (CMS).

Remarque
La version en ligne de cette présente notice est agrémentée de nombreux exemples, liens vers des ressources complémentaires, etc. Celle-ci est disponible à l'adresse : www.accede-web.com/notices/html-css-javascript/.

Contact

Pour toute remarque à propos de cette notice, merci de contacter Atalan, coordinateur du projet AcceDe Web à l'adresse suivante : accede@atalan.fr.

Vous pouvez également trouver plus d'informations sur les notices méthodologiques du projet AcceDe Web sur le site www.accede-web.com.

Licence d'utilisation

1. Structure générale

1.1. Structurer la zone d'entête principale avec <header role="banner">

La zone d'entête principale des pages, qui peut par exemple entre-autres contenir le logo et le moteur de recherche, doit être balisée avec `<header role="banner">`.

Attention
Tandis que la balise `<header>` peut être utilisée plusieurs fois dans une page web, `role="banner"` ne doit lui être employé qu'une seule fois.

Remarque
Il est tout à fait possible d'imbriquer plusieurs rôles ARIA :
`<div role="search">` dans `<header role="banner">`, par exemple.

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

1.2. Identifier le moteur de recherche avec role="search"

Le moteur de recherche du site doit être identifié avec `role="search"`.

`role="search"` doit être intégré dans la balise contenant le moteur de recherche (`<section>`, `<div>`, etc.) à l'exception de la balise `<form>`.

```
<div role="search">
  <form…>
    <input type="text" title="Recherche par mots-clés" />
    <input type="submit" value="Rechercher" />
  </form>
</div>
```

Attention
Une page ne doit contenir qu'un seul `role="search"`.

1.3. Structurer la zone de contenu principal avec <main role="main">

La zone de contenu principal des pages doit être balisée avec `<main role="main">`.

Attention
Une page ne doit contenir qu'une seule balise `<main>` et qu'un seul `role="main"`.

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

1.4. Identifier les informations relatives à l'auteur du site avec role="contentinfo"

Les informations relatives à l'auteur du site, comme par exemple le copyright ainsi que les liens « Mentions légales » et « Crédits », doivent être identifiées avec `role="contentinfo"`.

Attention
Une page ne doit contenir qu'un seul `role="contentinfo"`.

Remarque
Les référentiels d'accessibilité français (RGAA et AccessiWeb) imposent que `role="contentinfo"` soit intégré dans la balise `<footer>` de la zone de pied-de-page.

Bien que cela soit généralement parfaitement approprié, parfois, il se peut que les informations relatives à l'auteur du site figurent ailleurs qu'en pied de page. Dans ce cas, `role="contentinfo"` doit être intégré dans la balise entourant ces dernières.

```
<header role="banner">[…]</header>
<nav role="navigation">[…]</nav>
<main role="main">[…]</main>
<footer role="contentinfo">
   <ul>
      <li><a href="…">Plan du site</a></li>
      <li><a href="…">Crédits</a></li>
      <li><a href="…">Mentions légales</a></li>
   </ul>
   <p>© Mon site</p>
</footer>
```

1.5. Structurer les menus de navigation principaux et secondaires avec <nav role="navigation">

Les menus de navigation principaux et secondaires doivent être balisés avec `<nav role="navigation">`.

Remarque
La balise `<nav>` est à réserver uniquement pour englober des menus contenant des **liens internes** au site.

Autrement dit, elle ne doit pas être utilisée pour englober une liste de liens pointant vers des sites externes. Comme une liste de liens vers des réseaux sociaux, par exemple.

Astuce
Une bonne pratique consiste à ajouter l'attribut `aria-label` dans la balise `<nav>` et de le renseigner en précisant le type de système de navigation dont il est question.

```
<header role="banner">[…]</header>
<nav role="navigation" aria-label="Menu principal">[…]</nav>
<nav role="navigation" aria-label="Menu secondaire">[…]</nav>
<main role="main">
   <nav aria-label="Fil d'Ariane">[…]</nav>
   […]
</main>
<footer role="contentinfo">[…]</footer>
```

1.6. Structurer les menus de navigation avec des listes

Utiliser des listes non ordonnées `<ul>` et `<li>` pour baliser les menus de navigation.

Dans le cas d'un menu principal à plusieurs niveaux, veiller à la bonne imbrication des listes :

```
<nav role="navigation" aria-label="Menu principal">
 <ul>
  <li><a href="…">Premier lien du menu</a></li>
  <li>
    <a href="…">Deuxième lien du menu</a>
    <ul>
      <li><a href="…">Premier lien du sous-menu</a></li>
      <li><a href="…">Deuxième lien du sous-menu</a></li>
    </ul>
  </li>
  <li><a href="…">Troisième lien du menu</a></li>
 </ul>
</nav>
```

1.7. Mettre en place une hiérarchie de titres logique et exhaustive avec les balises <h1> à <h6>

Sur chaque page, pour baliser les titres, utiliser les balises de titres allant de `<h1>` jusqu'à `<h6>`. La structure des titres doit être à la fois logique et exhaustive.

C'est-à-dire :

- Qu'il ne doit pas y avoir de « sauts » ni d'incohérences dans la structure des titres (passage brutal d'un `<h1>` à un `<h3>` sans `<h2>` intermédiaire, par exemple).
- Que tous les éléments qui ont valeur de titres doivent être balisés comme tels.

Attention

En HTML5, il est possible de mettre en place une hiérarchie de titres complète, à plusieurs niveaux, en utilisant uniquement la balise `<h1>` et les balises destinées à créer des sections dans le document, comme `<article>` ou `<section>`.

Toutefois, il est préférable de continuer à structurer les titres de manière logique et exhaustive, comme cela était fait avant HTML5. C'est-à-dire en utilisant les titres de `<h1>` à `<h6>`.

Remarques

- Il n'est jamais gênant d'utiliser plusieurs `<h1>` dans une page si plusieurs titres de premier niveau sont présents.
- Une bonne pratique d'accessibilité consiste à ne pas ajouter de titres cachés.

Astuce

Pour mettre en place une hiérarchie de titres logique et exhaustive, il faut imaginer que les titres forment la « Table des matières » de la page. Est-elle logique ? Exhaustive ?

Remarque

Le role ARIA `heading` associé à un attribut `aria-level` (renseigné avec une valeur allant de 1 à 6) permet d'affecter la valeur d'un titre à n'importe quelle balise `<html>`.

Concrètement, par exemple :

- `<p role="heading" aria-level="1">Titre de niveau 1</p>` est sémantiquement équivalent à `<h1>Titre de niveau 1</h1>`.
- `<div role="heading" aria-level="3">Titre de niveau 3</div>` à `<h3>Titre de niveau 3</h3>`.

Cette technique n'étant toutefois pas optimale pour l'accessibilité, elle est à utiliser en ultime recours.

1.8. Écrire le code HTML en suivant la logique de l’ordre de lecture

L’ordre d’écriture des balises dans le flux HTML doit suivre la logique de l’ordre de lecture de la page.

C’est-à-dire que lorsqu’une balise précède immédiatement une autre balise lors du parcours visuel de la page, la première doit également précéder immédiatement la seconde dans le flux HTML.

Astuce
Pour tester cette recommandation, il suffit de désactiver les CSS est de s’assurer que le résultat obtenu correspond bien à l’ordre de lecture de la page, lorsque les CSS sont activées.

Attention
Si des contenus sont masqués par défaut, veiller à leur bon positionnement dans le flux HTML lorsque les styles sont désactivés.

1.9. Veiller à la cohérence de l’ordre du flux HTML d’une page à l’autre

L’ordre d’intégration des principaux blocs dans le flux HTML doit rester cohérent d’une page à l’autre du site.

Par exemple, si le menu secondaire est placé après le contenu principal, il est important de conserver cet ordre sur l’ensemble des pages du site.

2. Titre de la page

2.1. Renseigner un <title> précis sur chaque page

Sur chaque page, la balise `<title>` doit être renseignée avec précision.
Elle doit au minimum annoncer le nom de la page courante ainsi que le nom du site.

Astuces

- Il est recommandé de faire apparaître le nom de la page courante en premier dans la balise `<title>`. Il est par exemple possible d'opter pour `<title>[Nom de la page courante] | [Nom du site]</title>`.
- Une bonne pratique d'accessibilité consiste à veiller à la cohérence de l'ordre des contenus de la balise `<title>` sur l'ensemble des pages du site.

Remarque

Parfois, certaines pages sont rechargées avec un contenu modifié suite à une action de l'utilisateur.
C'est par exemple le cas :

- Lors de l'utilisation de filtres, comme des nuages de mots-clés.
- Lors de l'utilisation de la pagination.
- Lorsqu'une expression est recherchée via le formulaire de recherche.

Lorsque ces mises à jour se font côté serveur, il faut alors veiller à modifier le titre de la page en conséquence, par exemple :

```
<title>Résultats de votre recherche sur "[Expression recherchée]" (page 3/7) | [Nom du site]</title>
```

3. Langues

3.1. Renseigner la langue principale de la page avec l'attribut lang sur la balise <html>

Afin de garantir la bonne restitution des contenus textuels à l'utilisateur, une déclaration de langue principale doit être effectuée sur chaque page.

Utiliser pour cela l'attribut `lang` sur la balise `<html>`.

Par exemple, pour une page en français :

```
<html lang="fr">
```

Astuce

Pour renseigner l'attribut `lang`, un code de langue sur deux lettres (ou, s'il n'est pas disponible, sur trois lettres) doit être utilisé.

Les principaux codes utilisés sont les suivants :

- `fr` pour le français.
- `en` pour l'anglais.
- `es` pour l'espagnol.
- `de` pour l'allemand.
- `it` pour l'italien.

3.2. Utiliser l'attribut lang pour signaler les changements de langue

Si des contenus sont proposés dans une langue différente de la langue principale, alors ils doivent être signalés avec l'attribut `lang`.

Par exemple, dans le cas d'une page en français :

```
<a href="…" lang="en" hreflang="en">English version</a>
```

Astuce

Si aucune balise n'encadre directement le contenu en langue étrangère, alors utiliser la balise `<span>` ou `<div>` et renseigner son attribut `lang`.

Remarque

Signaler le changement de langue n’est pas demandé pour :

- Les noms propres.
- Les mots d’origine étrangère intégrés dans le dictionnaire de la langue principale.
- Tous les mots d’origine étrangère mais qui se prononcent et se comprennent correctement avec l’accent de la langue principale (« podcast », par exemple, si la langue principale est le français).

4. Grammaire HTML et sémantique

4.1. Écrire un code HTML valide selon les règles de grammaire du DOCTYPE utilisé

Sur chaque page, un DOCTYPE valide doit être utilisé et le code HTML doit être conforme aux règles de grammaire de ce dernier (le choix du DOCTYPE est libre).

Remarque
Ce DOCTYPE doit impérativement être intégré avant la balise `<html>`.

Concernant les règles de grammaire, il est particulièrement important de veiller à :

- La correcte imbrication des balises.
- La correcte ouverture et fermeture des balises.
- L'absence d'attributs dupliqués sur une même balise.
- L'unicité des valeurs de l'attribut id au sein d'une même page.

Remarque
Les balises et attributs HTML obsolètes et/ou destinés exclusivement à la mise en forme ne doivent pas être utilisés.

4.2. Employer les balises HTML pour leur valeur sémantique

Les balises HTML doivent être utilisées pour leur valeur sémantique et non pour leur rendu visuel.

Balises	Mauvais usage	Bon usage
`<a>`	Obtenir un bouton d'action sans arrière-plan et bordure	Baliser un lien hypertexte
`<hr />`	Obtenir une ligne séparatrice	Séparer des sujets différents dans un bloc de texte
`<fieldset>`	Obtenir une bordure	Regrouper des éléments de formulaires de même nature
`<q>`	Obtenir un texte entre guillemets	Baliser une citation courte

5. Liens et boutons

5.1. Différencier les boutons des liens

Pour une meilleure compréhension des interfaces riches par les utilisateurs, réserver les boutons pour les actions et les liens pour la navigation :

Boutons

Les boutons `<button>` / `<input />` permettent à l'utilisateur de déclencher des actions.

Ils sont par exemple à employer pour soumettre des informations, afficher l'élément suivant ou précédent dans un carrousel, fermer une fenêtre modale, etc.

Liens

Les liens `<a>` permettent à l'utilisateur de naviguer.

Ils sont à employer pour pointer sur une autre page ou sur une zone précise de la page courante par l'intermédiaire d'un système d'ancre.

5.2. Compléter les liens et les boutons non explicites avec du texte caché, aria-label ou title

Un lien ou un bouton considéré comme non explicite est un lien ou un bouton dont l'intitulé seul ne permet pas de comprendre sa destination ou sa fonction.

- Les liens « Lire la suite », « En savoir plus », « Plus d'informations », « Cliquer ici » sont par exemple considérés comme non explicites par nature.
- Idem pour le bouton « Ok ».

Dans ces situations, du texte caché, l'attribut `aria-label` ou l'attribut `title` doit alors être utilisé pour préciser la destination du lien ou la fonction du bouton.

Prenons ci-dessous l'exemple d'un lien « Lire la suite » pointant vers la page du projet AcceDe Web pour présenter ces trois techniques.

Texte caché

Rendre un lien ou un bouton explicite via du texte caché consiste à :

1. Ajouter une balise `<span>` dans la balise `<a>` ou `<button>`.
2. Intégrer les informations permettant de rendre explicite le lien ou le bouton dans cette balise `<span>`.

3. Masquer le contenu de ce `<span>` en CSS en le rendant invisible ou en le sortant de l'écran (hors Viewport) respectivement à l'aide des propriétés CSS `opacity: 0;` ou `position: absolute;`.

```
<a href="…">
   <span class="hors-ecran">Le projet AcceDe Web</span>
   Lire la suite
</a>
```

```
.hors-ecran {
   position: absolute;
   left: -99999px;
}
```

Attention

Le contenu du `<span>` ne doit pas être masqué avec les propriétés CSS « display: none; » ou « `visibility: hidden;` ».

Remarque

La technique du texte caché ne convient pas pour rendre explicite un bouton de type `<input />`.

Attribut `aria-label`

Rendre un lien ou un bouton explicite via l'attribut `aria-label` consiste à :

1. Ajouter l'attribut aria-label dans la balise `<a>`, `<button>` ou `<input />`.
2. Renseigner cet attribut avec les informations permettant de rendre explicite le lien ou le bouton puis en reprenant à l'identique la valeur de l'intitulé du lien ou du bouton lui-même.

```
<a href="…" aria-label="Le projet AcceDe Web (Lire la suite)">
   Lire la suite
</a>
```

Astuce

Une bonne pratique consiste à renseigner cet attribut en commençant par l'information la plus explicite puis en terminant par la reprise de l'intitulé.

Attribut `title`

Rendre un lien ou un bouton explicite via l'attribut `title` consiste à :

1. Ajouter l'attribut `title` dans la balise `<a>`, `<button>` ou `<input />`.
2. Renseigner cet attribut avec les informations permettant de rendre explicite le lien ou le bouton puis en reprenant à l'identique la valeur de l'intitulé du lien ou du bouton lui-même.

```
<a href="…" title="Le projet AcceDe Web (Lire la suite)">
   Lire la suite
</a>
```

Astuce
Une bonne pratique consiste à renseigner cet attribut en commençant par l'information la plus explicite puis en terminant par la reprise de l'intitulé.

Remarque
Une de ces trois techniques doit également être utilisée pour distinguer les liens ou boutons dont les intitulés pourraient être considérés comme explicites, mais qui pointent vers des pages ou déclenchent des actions différentes.

Par exemple, dans le cas de deux boutons « Rechercher » affichés sur la même page, et qui seraient à distinguer :

```
<input type="submit" value="Rechercher" title="Rechercher une actualité" />
[…]
<input type="submit" value="Rechercher" title="Rechercher une personne dans l'annuaire" />
```

5.3. Ne pas utiliser les attributs title et aria-label sur des liens ou boutons explicites

Un lien ou un bouton considéré comme explicite est un lien ou un bouton dont l'intitulé seul permet de comprendre sa destination ou sa fonction.

Par exemple :

- Le lien « Le projet AcceDe Web » est explicite par nature.
- Idem pour le bouton « Valider ma commande ».

Les attributs `title` et `aria-label` (qui peuvent servir à rendre accessibles des liens non explicites) ne doivent pas être employés pour ce type de liens.

Voici concrètement quelques exemples d'usages **incorrects** de l'attribut `title` dans une balise `<a>` :

```
<a href="…" title="">Le projet AcceDe Web</a>
```

```
<a href="…" title="Le projet AcceDe Web">Le projet AcceDe Web</a>
```

```
<a href="…" title="AcceDe Web">Le projet AcceDe Web</a>
```

Et quelques exemples d'usages **incorrects** de l'attribut `aria-label` dans une balise `<input />` :

```
<input type="submit" value="Valider ma commande" aria-label="" />
```

```
<input type="submit" value="Valider ma commande" aria-label="Valider ma commande" />
```

5.4. Intégrer le poids et le format des fichiers dans chaque lien et bouton permettant de les télécharger

Chaque fois qu'un lien ou un bouton pointe sur un fichier à télécharger, les informations suivantes doivent être intégrées dans l'intitulé :

- Le nom du document.
- Le format du document.
- Le poids du document.

```
<a href="plan-reseau-bus-lyon.pdf">
Télécharger le plan du réseau de bus de Lyon (PDF - 2 Mo)
</a>
```

Remarque

Dans le cas où le format du fichier à télécharger est indiqué via un pictogramme, se référer à la thématique « 6. Images et icônes » pour l'intégrer de manière accessible.

Astuce
Dans le cas où le poids du fichier à télécharger ne peut être connu à l'avance (génération à la volée, par exemple), une bonne pratique d'accessibilité consiste à indiquer un poids approximatif ou maximal.

Pour obtenir par exemple :

```
<a href="facture.pdf">
Télécharger votre facture du mois de mars (PDF - environ 1,5 Mo)
</a>
```

5.5. Signaler l'ouverture des nouvelles fenêtres

Chaque fois qu'un lien ou un bouton déclenche l'ouverture d'une nouvelle fenêtre dans le navigateur, l'utilisateur doit être prévenu par l'ajout d'une mention du type « (nouvelle fenêtre) ».

Soit directement dans l'intitulé :

```
<a href="cgv.html" target="_blank">
   Conditions Générales de Vente (nouvelle fenêtre)
</a>
```

Soit dans l'attribut `title` :

```
<a href="cgv.html" target="_blank" title="Conditions Générales de Vente
(nouvelle fenêtre)">
   Conditions Générales de Vente
</a>
```

Soit via le texte de remplacement d'une icône/image, sous la forme :

- D'une police d'icônes (Icon Fonts) – *voir recommandation 6.1*.
- D'un svg (image vectorielle) – *voir recommandation 6.2*.
- D'une balise <img /> – *voir recommandation 6.3*.

Astuce
À noter qu'il est également conforme de passer par l'attribut `aria-label` :

```
<a href="cgv.html" target="_blank" aria-label="Conditions Générales de Vente
(nouvelle fenêtre)">
   Conditions Générales de Vente
</a>
```

6. Images et icônes

6.1. Polices d'icônes (Icon Fonts)

Remarque
Que l'icône soit décorative, informative ou qu'il s'agisse d'un lien/bouton, la balise servant à l'afficher doit, dans tous les cas, intégrer `aria-hidden="true"` afin de s'assurer que les lecteurs d'écran ne restituent aucun contenu à l'utilisateur lorsqu'elle est parcourue.

Aussi, idéalement, la balise servant à afficher l'icône doit être un `<span>` car sans valeur sémantique.

Icônes décoratives/illustratives

Lorsqu'une icône décorative ou illustrative est intégrée dans le code HTML, rien n'est à prévoir.

Dans l'exemple de code HTML ci-après, l'icône accompagne simplement le titre « Messages d'erreur » (explicite par nature) :

```
<h2>
   <span class="icone erreur" aria-hidden="true"></span>
   Messages d'erreur
</h2>
```

Icônes informatives

Lorsqu'une icône informative est intégrée dans le code HTML :

1. Ajouter une balise (`<span>`, par exemple) juste après la balise servant à afficher l'icône.
2. Intégrer les informations permettant de rendre explicite l'icône dans cette nouvelle balise.
3. Masquer le contenu de cette balise en CSS en le rendant invisible ou en le sortant de l'écran (hors Viewport) respectivement à l'aide des propriétés CSS « `opacity: 0;` » ou « `position: absolute;` ».

Dans l'exemple de code HTML ci-après, l'icône indique que le lien « Conditions Générales de Vente » s'ouvre dans une nouvelle fenêtre :

```
<h1>
   Conditions Générales de Vente
   <span class="icone nouvelle-fenetre" title="(nouvelle fenêtre)" aria-
hidden="true"></span>
   <span class="hors-ecran">(nouvelle fenêtre)</span>
</h1>
```

```
.hors-ecran {
   position: absolute;
   left: -99999px;
}
```

Astuce

Une bonne pratique consiste à intégrer un attribut `title` reprenant la valeur du texte de remplacement dans la balise servant à afficher l'icône.

Icônes liens ou boutons seules

Lorsqu'une icône seule (sans intitulé) servant de lien ou de bouton est intégrée dans le code HTML :

1. Ajouter une balise `<span>` dans la balise `<a>` ou `<button>`.
2. Intégrer les informations permettant de rendre explicite le lien ou le bouton dans ce `<span>`.
3. Masquer le contenu de ce `<span>` en CSS en le rendant invisible ou en le sortant de l'écran (hors Viewport) respectivement à l'aide des propriétés CSS « `opacity: 0;` » ou « `position: absolute;` ».

Dans l'exemple de code HTML ci-après, l'icône-lien pointe vers la page d'accueil d'un site :

```
<a href="/">
   <span class="icone accueil" title="Accueil" aria-hidden="true"></span>
   <span class="hors-ecran">Accueil</span>
</a>
```

```
.hors-ecran {
   position: absolute;
   left: -99999px;
}
```

Astuce
Une bonne pratique consiste à intégrer un attribut `title` reprenant la valeur du texte de remplacement dans la balise servant à afficher l'icône.

6.2. svg (images vectorielles)

<svg> décoratifs/illustratifs

Lorsqu'un `<svg>` est simplement décoratif ou illustratif, lui intégrer `aria-hidden="true"`.

Dans l'exemple de code HTML ci-après, le `<svg>` accompagne simplement le titre « Messages d'erreur » (explicite par nature) :

```
<h2>
   <svg aria-hidden="true">[…]</svg>
   Messages d'erreur
</h2>
```

Attention
Un `<svg>` simplement décoratif ou illustratif ne doit pas posséder d'attributs `title`, `aria-label`, `aria-labelledby` et/ou `aria-describedby`.

De la même manière, il ne doit pas posséder de balises `<title>` et/ou `<desc>` renseignées.

<svg> informatifs

Lorsqu'un `<svg>` informatif est intégré dans le code HTML :

- L'englober d'une balise de type block (`<div>`, par exemple).
- Intégrer `aria-hidden="true"` dans ses balises enfants `<g>` et `<path>`.
- Ajouter une balise `<text>` dans le `<svg>` et la renseigner avec une information égale ou équivalente à celle véhiculée par l'image.
- Cacher ce texte de remplacement en intégrant `opacity="0"` dans la balise `<text>`.

Dans l'exemple de code HTML ci-après, le `<svg>` indique que le lien « Conditions Générales de Vente » s'ouvre dans une nouvelle fenêtre :

```
<h1>
   Conditions Générales de Vente
   <svg …>
      <text opacity="0" …>(nouvelle fenêtre)</text>
      <g aria-hidden="true">
         <path aria-hidden="true" …></path>
      </g>
   </svg>
</h1>
```

Attention

Il est fortement déconseillé de rédiger le texte de remplacement d'un `<svg>` en commençant par exemple par la mention « Image […] ».

`<svg>` liens ou boutons seuls

- Lorsqu'un `<svg>` seul (sans intitulé) servant de lien ou de bouton est intégré dans le code HTML :
- L'englober d'une balise de type block (`<div>`, par exemple).
- Intégrer `aria-hidden="true"` dans ses balises enfants `<g>` et `<path>`.
- Ajouter une balise `<text>` dans le `<svg>` et la renseigner avec les informations permettant de rendre le lien ou le bouton explicite.
- Cacher ce texte de remplacement en intégrant `opacity="0"` dans la balise `<text>`.
- Intégrer `role="link"` dans la balise `<svg>` si celle-ci sert de lien.
- Intégrer `role="button"` dans la balise `<svg>` si celle-ci sert de bouton.

Dans l'exemple de code HTML ci-après, le `<svg>` pointe vers la page d'accueil d'un site :

```
<div>
   <svg role="link" …>
      <a …>
         <text opacity="0" …>Accueil</text>
      </a>
      <g aria-hidden="true">
         <path aria-hidden="true" …></path>
      </g>
   </svg>
</div>
```

Attention
Il est fortement déconseillé de rédiger le texte de remplacement d'un `<svg>` servant de lien ou bouton en commençant respectivement par « Lien vers […] » ou « Bouton […] ». Cette information sera déjà annoncée par les aides techniques respectivement via `role="link"` et `role="button"`.

Remarque
Bien que les techniques présentées sur cette fiche fonctionnent sur **le plus grand nombre de combinaisons de navigateurs/lecteurs d'écran**, elles ne sont pas conformes à ce qui est demandé dans les référentiels français d'accessibilité (RGAA 3.0 et AccessiWeb HTML5/ARIA).

Pour obtenir une conformité, il est important de se référer directement à ces référentiels.

6.3. Balises <img /> et <input type="image" />

Remarque
Que la balise `<img />` soit décorative, informative ou qu'elle serve de lien/bouton, elle doit dans tous les cas posséder un attribut `alt` ; ce afin de s'assurer que les lecteurs d'écran ne restituent pas sa source (soit le contenu de son attribut `src`).

`<img />` décoratives/illustratives

Lorsqu'une balise `<img />` décorative ou illustrative est intégrée dans le code HTML, l'attribut `alt` doit être laissé vide (sans aucun espace entre les guillemets de `alt=""`).

Dans l'exemple de code HTML ci-après, la balise `<img />` accompagne simplement le titre « Messages d'erreur » (explicite par nature) :

```
<h2>
   <img src="erreur.png" alt="" />
   Messages d'erreur
</h2>
```

`<img />` informatives

Lorsqu'une balise `<img />` informative est intégrée dans le code HTML, renseigner son attribut `alt` avec une information égale ou équivalente à celle véhiculée par l'image.

Dans l'exemple de code HTML ci-après, la balise `<img />` indique que le lien « Conditions Générales de Vente » s'ouvre dans une nouvelle fenêtre :

```
<h1>
   Conditions Générales de Vente
   <img src="nouvelle-fenetre.png" alt="(nouvelle fenêtre)" />
</h1>
```

Attention

Il est fortement déconseillé de rédiger le texte de remplacement d'une image en commençant par `alt="Image […]"`. Cette information sera déjà annoncée par les aides techniques lors de la lecture d'une balise `<img />`.

`<img />` liens ou boutons seules

Lorsqu'une balise `<img />` seule (sans intitulé) servant de lien ou de bouton est intégrée dans le code HTML, intégrer les informations permettant de le rendre explicite dans son attribut `alt`.

Dans l'exemple de code HTML ci-après, la balise `<img />` pointe vers la page d'accueil d'un site :

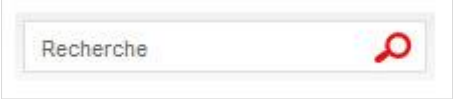
```
<a href="/">
   <img src="accueil.png" alt="Accueil" />
</a>
```

Attention

Il est fortement déconseillé de rédiger l'attribut `alt` d'une balise `<img />` servant de lien ou bouton en commençant respectivement par `alt="Lien vers […]"` ou `alt="Bouton […]"`. Cette information sera déjà annoncée par les aides techniques lors de la lecture d'une balise `<a>` ou `<button>`.

`<input type="image" />`

Lorsqu'une balise `<input type="image" />` est intégrée dans le code HTML, intégrer les informations permettant de rendre le bouton explicite dans son attribut `alt`.



Dans l'exemple de code HTML ci-après, la balise `<input type="image" />` permet de lancer une recherche dans un site :

```
<input type="image" src="loupe.png" alt="Lancer la recherche"  />
```

Attention
Il est fortement déconseillé de rédiger l'attribut alt d'une balise `<input type="image" />` en commençant par `alt="Bouton […]"`. Cette information sera déjà annoncée par les aides techniques lors de la lecture d'une balise `<input type="image" />`.

6.4. Baliser les images légendées avec <figure role="group"> et <figcaption>

Les images légendées doivent être balisées avec `<figure role="group">` et `<figcaption>`.

La balise `<figure role="group">` doit englober l'image ainsi que la légende, qui doit de son côté être balisée avec `<figcaption>`.

Remarque
À noter que le texte de remplacement de l'image (l'attribut `alt` pour la balise `<img />`, par exemple) doit contenir une référence à la légende adjacente.

Exemple

Photo 1 : prise de vue par Audesou en février 2013.

```
<figure role="group">
   <img src="…" alt="Trois lamas dans le désert en Bolivie. [photo 1]" />
   <figcaption>
      Photo 1 : prise de vue par Audesou en février 2013.
   </figcaption>
</figure>
```

6.5. Ne pas utiliser CSS pour afficher les images porteuses d'informations

Les CSS ne doivent pas être utilisées pour afficher les images porteuses d'informations.

En d'autres termes, chaque fois qu'une image apporte de l'information, elle doit être intégrée en dur dans le code HTML (via balise `<img />`, `svg` ou polices d'icônes, par exemple).

Remarque
L'usage de sprites (images chargées en arrière-plan via CSS) n'est donc pas autorisé pour les images porteuses d'informations.

Astuce
Sont considérées comme porteuses d'informations toutes les images qui entraîneraient une perte de contenus ou de fonctionnalités si elles n'étaient pas disponibles :

- Logos.
- Images-textes.
- Images-liens et boutons.
- Etc.

Attention
En particulier, la technique qui consiste à sortir un texte de l'écran pour le remplacer par une image d'arrière-plan ne doit pas être utilisée.

Lorsque les images ne sont pas chargées l'information peut être perdue.

```
<a href="/" id="logo">[Le texte de mon logo]</a>
```

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

Cette technique est à remplacer par l'utilisation de HTML pur.

7. Formulaires

7.1. Utiliser la balise <label> ainsi que les attributs for et id pour étiqueter les champs avec intitulé visible

Pour étiqueter les champs disposant d'un intitulé visible :

1. Utiliser `<label>` pour baliser chaque intitulé.
2. Ajouter un attribut `for` sur chaque balise `<label>` ainsi qu'un attribut id sur chaque champ.
3. Renseigner avec une valeur unique et identique les attributs for et id de chaque couple intitulé/champ.

```
<label for="nom">Votre nom</label>
<input type="text" id="nom" name="nom" />
```

```
<label for="pays">Votre pays</label>
<select id="pays" name="pays">
   <option value="belgique">Belgique</option>
   <option value="france">France</option>
   […]
</select>
```

Attention

Les attributs `title` et `aria-label` ne doivent jamais être utilisés pour les champs étiquetés de cette manière.

Remarque

Il est important de prévoir des intitulés identiques pour les champs dont la fonction est identique.

Par exemple, si plusieurs formulaires d'identification sont présents dans le site, ne pas utiliser l'intitulé « Identifiant » pour l'un et « Login » pour l'autre.

7.2. Utiliser title ou aria-label pour étiqueter les champs sans intitulé visible

Pour étiqueter les champs sans intitulé visible :

1. Intégrer un attribut `title` ou `aria-label` dans le champ.
2. Renseigner cet attribut en précisant la fonction du champ.

```
<input type="text" title="Votre recherche" name="recherche" />
```

```
<select title="Filtrer les actualités" name="filtre">
   <option>Filtrer les actualités</option>
   <option>Par date de publication</option>
   <option>Par thématique</option>
   […]
</select>
```

Remarque

Il est également possible d'étiqueter les champs sans intitulé visible via la balise `<label>` et les attributs for et id.

Il s'agit alors de masquer la balise <label> en la sortant de l'écran.
Par exemple via les déclarations CSS « `position: absolute;` » et « `left: -99999px;` ».

Attention

L'attribut `placeholder` ne peut faire office d'étiquette.

Il est uniquement à réserver pour des aides à la saisie secondaires.

```
<input type="text" title="Votre recherche" placeholder="Saisissez vos mots-clés..." />
```

7.3. Intégrer les aides à la saisie directement dans les balises <label>

Les aides à la saisie doivent être intégrées directement dans les balises `<label>`.

C'est notamment le cas :

- Des indications qui permettent de connaître le format de saisie attendu, du type « jj/mm/aaaa » pour un format de date.
- Des indications qui permettent de connaître le type et le poids maximal autorisé pour l'envoi de fichiers.
- Des mentions du type « Champ obligatoire ».
- Etc.

```
<label for="date">
   Votre date de naissance
   <span>(au format jj/mm/aaaa)</span>
</label>
<input type="text" id="date" name="date" />
```

```
<label for="numero">
    Votre numéro de client
    <input type="text" id="numero" name="numero" />
    <span>Par exemple : 76432-BT-VZ</span>
</label>
```

Remarque

Dans certains cas, pour des raisons de mise en page particulière par exemple, il n'est pas possible d'intégrer l'aide à la saisie directement dans la balise <label>.

Dans ce cas-là :

- Intégrer un attribut `id` dans la balise englobant le message d'erreur.
- Renseigner cet attribut `id` avec une valeur unique.
- Intégrer l'attribut `aria-describedby` dans le champ correspondant.
- Renseigner cet attribut `aria-describedby` en reprenant la valeur de l'attribut `id` du message d'erreur.

```
<label for="document">
   Ajouter un document à votre dossier
</label>
<input type="file" id="document" name="document" aria-describedby="formats"
/>
<h2>Documents actuellement dans votre dossier</h2>
<ul>
   <li>CV</li>
   <li>Lettre de motivation</li>
</ul>
<p id="formats">Formats acceptés : pdf ou doc.</p>
```

À noter que l'attribut `aria-describedby` peut recevoir plusieurs `id`.

Attention

L'attribut `placeholder` ne doit pas être utilisé pour les aides à la saisie nécessaires à la compréhension des données attendues.

Il est uniquement à réserver pour des aides à la saisie secondaires.

```
<input type="text" title="Votre recherche" placeholder="Saisissez vos mots-
clés..." />
```

7.4. Intégrer required ou aria-required="true" dans les champs obligatoires

Les champs obligatoires doivent intégrer `required` ou `aria-required="true"`.

Remarque
En complément de cet attribut, un signe distinctif doit être intégré dans la balise `<label>`.

```
<label for="email">Votre email *</label>
<input type="text" id="email" name="email" required />
<input type="checkbox" id="conditions" aria-required="true" />
<label for="conditions">J'ai lu et j'accepte les conditions générales de
vente (obligatoire)</label>
```

7.5. Intégrer les messages d'erreurs et les suggestions de correction directement dans les balises <label>

Dans le cas où les messages d'erreurs et les suggestions de correction sont positionnés au niveau de chaque champ concerné, alors les intégrer directement dans les balises `<label>`.

Astuce
En complément du message d'erreur, une bonne pratique d'accessibilité consiste à ajouter `aria-invalid="true"` dans le champ.

```
<label for="nom">
   Votre nom *
   <span>Veuillez renseigner votre nom</span>
</label>
<input type="text" id="nom" name="nom" required aria-invalid="true" />
```

```
<label for="courriel">
   Votre courriel *
   <input type="text" id="courriel" name="courriel" required aria-
invalid="true" />
   <span>Veuillez respecter le format du courriel (exemple@domaine.fr)</span>
</label>
```

Remarque
Dans certains cas, pour des raisons de mise en page particulière par exemple, il n'est pas possible d'intégrer l'aide à la saisie directement dans la balise <label>.

Dans ce cas-là :

- Intégrer un attribut `id` dans la balise englobant le message d'erreur.
- Renseigner cet attribut `id` avec une valeur unique.

- Intégrer l'attribut `aria-describedby` dans le champ correspondant.
- Renseigner cet attribut `aria-describedby` en reprenant la valeur de l'attribut `id` du message d'erreur.

```
<label for="document">
   Ajouter un document à votre dossier
</label>
<input type="file" id="document" name="document" aria-invalid="true" aria-describedby="formats erreur" />
<h2>Documents actuellement dans votre dossier</h2>
<ul>
   <li>CV</li>
   <li>Lettre de motivation</li>
</ul>
<p id="erreur">Format de fichier incorrect.</p>
<p id="formats">Formats acceptés : pdf ou doc.</p>
```

À noter que l'attribut `aria-describedby` peut recevoir plusieurs `id`.

7.6. Mettre à jour le <title> de la page quand celle-ci se recharge pour afficher une erreur ou un message de confirmation

Chaque fois qu'un formulaire renvoie une erreur ou un message de confirmation **suite à un rechargement de la page**, la balise `<title>` doit être mise à jour.

Par exemple, dans le cas d'une réussite :

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

Et dans le cas d'une erreur :

```
<title>Erreur - Contact | [Nom du site]</title>
```

Astuce
Il est recommandé de faire apparaître cette mention en premier dans la balise `<title>`.

Remarque

Dans certaines situations, il n'est pas nécessaire d'ajouter une mention car le titre de la page affichée après la soumission rend le résultat de l'action évident.

Par exemple :

- Formulaire de connexion qui envoie vers une page « Profil utilisateur ».
- Bouton « Passer à l'étape suivante » qui envoie à l'étape suivante dans un formulaire à étapes multiples.
- Formulaire de contact qui envoie une page de prévisualisation.

7.7. Regrouper et titrer les champs de même nature avec <fieldset> et <legend>

Lorsqu'un formulaire propose plusieurs groupes de champs dont certains ont des intitulés identiques, utiliser les balises `<fieldset>` et `<legend>`.

```
<fieldset>
   <legend>Participant 1</legend>
   <label for="prenom-1">Prénom</label>
   <input type="text" id="prenom-1" name="prenom-1" />
   <label for="nom-1">Nom</label>
   <input type="text" id="nom-1" name="nom-1" />
   […]
</fieldset>

<fieldset>
   <legend>Participant 2</legend>
   <label for="prenom-2">Prénom</label>
   <input type="text" id="prenom-2" name="prenom-2" />
   <label for="nom-2">Nom</label>
   <input type="text" id="nom-2" name="nom-2" />
   […]
</fieldset>
```

Attention

Les balises `<fieldset>` et `<legend>` ne sont à utiliser que lorsque plusieurs groupes de champs disposent d'intitulés identiques.

Par exemple :

- Une série de questions sur une même page avec comme réponses possibles « oui » ou « non ».
- Une liste de participants à un évènement avec à chaque fois « nom » et « prénom ».

Dans toutes les autres situations :

- Ne pas utiliser les balises `<fieldset>` et `<legend>`.
- Utiliser les balises `<h1>` à `<h6>` pour titrer les groupes de champs.

Remarque

Une bonne pratique d'accessibilité consiste à utiliser également les balises `<fieldset>` et `<legend>` lors de l'intégration de listes de boutons radio ou de cases à cocher dans la page.

Par exemple :

```
<fieldset>
  <legend>Sports pratiqués</legend>
  <ul>
    <li>
      <input type="checkbox" id="basket" />
      <label for="basket">Basket</label>
    </li>
    <li>
      <input type="checkbox" id="tennis" />
      <label for="tennis">Tennis</label>
    </li>
    […]
  </ul>
</fieldset>
```

7.8. Ordonner les options de manière logique dans les listes déroulantes

Lorsque des listes déroulantes sont utilisées, les options qu'elles contiennent doivent être ordonnées de manière logique.

L'ordre logique dépend du contexte. Il peut s'agir notamment :

- D'un ordre alphabétique (une liste de langues, par exemple).
- D'un ordre numérique (une liste de départements, par exemple).
- D'un ordre pratique (« France » en premier dans un formulaire d'inscription à un service français).

Astuce

Une bonne pratique d'accessibilité consiste à ne pas préfixer le contenu de la balise `<option>` avec des caractères décoratifs (tirets, étoiles, espaces, etc.).

Ceci afin de permettre un accès direct à une valeur ou à un groupe de valeurs souhaité par simple pression d'une touche au clavier (pression sur la touche « I » pour atteindre le pays « Italie », par exemple).

```
<select>
    <option>--Allemagne</option>
    <option>--Belgique</option>
    <option>--Espagne</option>
    <option>--France</option>
```

```
      <option>--Italie</option>
   </select>
```

Sera remplacé avantageusement par :

```
<select>
   <option>Allemagne</option>
   <option>Belgique</option>
   <option>Espagne</option>
   <option>France</option>
   <option>Italie</option>
</select>
```

8. Listes

8.1. Baliser les listes non ordonnées avec <ul> et

Utiliser les balises `<ul>` et `<li>` pour baliser les listes d'éléments pour lesquelles l'ordre n'a pas d'importance (menus, onglets, boutons de partage, plan du site, etc.).

Le cas échéant, veiller à la bonne imbrication des listes :

```
<ul>
   <li>Premier item de premier niveau</li>
   <li>
      Deuxième item de premier niveau
      <ul>
         <li>Premier item de deuxième niveau</li>
         <li>Deuxième item de deuxième niveau</li>
      </ul>
   </li>
   <li>Troisième item de premier niveau</li>
</ul>
```

8.2. Baliser les listes ordonnées avec <ol> et

Utiliser les balises `<ol>` et `<li>` pour baliser les listes d'éléments pour lesquelles l'ordre a de l'importance (liste d'étapes dans un processus, classement quelconque, etc.). C'est-à-dire lorsque l'information ne serait plus comprise si les éléments étaient placés dans un ordre différent.

Le cas échéant, veiller à la bonne imbrication des listes :

```
<ol>
   <li>Premier item de premier niveau</li>
   <li>
      Deuxième item de premier niveau
      <ul>
         <li>Premier item de deuxième niveau</li>
         <li>Deuxième item de deuxième niveau</li>
      </ul>
   </li>
   <li>Troisième item de premier niveau</li>
</ol>
```

8.3. Baliser les listes de descriptions avec <dl>, <dt> et <dd>

Utiliser les balises `<dl>`, `<dt>` et `<dd>` pour baliser les listes de descriptions.

Une liste de descriptions correspond à un groupement de clés/valeurs que l'on peut rencontrer par exemple sur une fiche produit ou un glossaire.

Remarque

Une clé (balise `<dt>`) peut avoir plusieurs valeurs (balise `<dd>`).

Exemple avec une description d'un événement :

```
<h2>Conférence sur l'accessibilité web</h2>
<dl>
   <dt>Lieu</dt>
   <dd>Paris</dd>
   <dt>Dates</dt>
   <dd>Samedi 7 septembre</dd>
   <dd>Mercredi 14 octobre</dd>
   <dt>Heure</dt>
   <dd>À partir de 10 heures</dd>
</dl>
```

Exemple avec un glossaire :

```
<dl>
   <dt><dfn>ARIA</dfn></dt>
   <dd lang="en">Accessible Rich Internet Application</dd>
   […]
   <dt><dfn>RGAA</dfn></dt>
   <dd>Référentiel Général d'Accessibilité pour les Administrations</dd>
   […]
</dl>
```

9. Tableaux

9.1. Donner un titre à chaque tableau de données avec la balise <caption>

Chaque fois qu'un tableau de données est intégré dans la page HTML, celui-ci doit posséder un titre clair et concis.

Ce titre doit être balisé avec `<caption>` et doit introduire le contenu du tableau.

```
<table>
   <caption>Températures moyennes mensuelles des 3 plus grandes villes de
France.</caption>
   […]
</table>
```

9.2. Baliser chaque cellule d'entête de ligne et de colonne avec <th>

Dans chaque tableau de données, baliser chaque cellule d'entête de ligne et de colonne avec la balise `<th>`.

C'est-à-dire que chaque fois qu'une cellule est nécessaire à la compréhension des données proposées dans le tableau, celle-ci doit être balisée avec `<th>`.

```
<table>
   <caption>Températures moyennes mensuelles des 3 plus grandes villes de
France.</caption>
   <tr>
      <td> </td>
      <th>Paris</th>
      <th>Marseille</th>
      <th>Lyon</th>
   </tr>
   <tr>
      <th>Juin</th>
      <td>22°C</td>
      <td>28°C</td>
      <td>26°C</td>
   </tr>
   <tr>
      <th>Juillet</th>
      <td>24°C</td>
      <td>30°C</td>
      <td>28°C</td>
   </tr>
</table>
```

Températures moyennes mensuelles des 3 plus grandes villes de France.

	Paris	Marseille	Lyon
Juin	22°C	28°C	26°C
Juillet	24°C	30°C	28°C

9.3. Utiliser l'attribut scope pour associer les cellules aux entêtes des tableaux de données simples

Un tableau de données simple est un tableau dans lequel les cellules d'entêtes s'appliquent systématiquement à la totalité des cellules de données d'une ligne ou d'une colonne.

Pour associer les entêtes à leurs données dans ce type de tableaux, utiliser l'attribut `scope` sur les balises `<th>`. La valeur de cet attribut changera selon que la cellule d'entête concerne :

- La totalité d'une colonne : `scope="col"`.
- La totalité d'une ligne : `scope="row"`.

```
<table>
   <caption>Températures moyennes mensuelles des 3 plus grandes villes de
France.</caption>
   <tr>
      <td> </td>
      <th scope="col">Paris</th>
      <th scope="col">Marseille</th>
      <th scope="col">Lyon</th>
   </tr>
   <tr>
      <th scope="row">Juin</th>
      <td>22°C</td>
      <td>28°C</td>
      <td>26°C</td>
   </tr>
   <tr>
      <th scope="row">Juillet</th>
      <td>24°C</td>
      <td>30°C</td>
      <td>28°C</td>
   </tr>
</table>
```

Températures moyennes mensuelles des 3 plus grandes villes de France.

	Paris	Marseille	Lyon
Juin	22°C	28°C	26°C
Juillet	24°C	30°C	28°C

9.4. Utiliser les attributs headers et id pour associer les cellules aux entêtes des tableaux de données complexes

Un tableau de données complexe est un tableau dans lequel les cellules d'entêtes ne s'appliquent pas systématiquement à la totalité des cellules de données d'une ligne ou d'une colonne.

Pour associer les entêtes aux données dans ce type de tableaux, utiliser les attributs `id` (identifiants) sur les cellules `<th>` et `headers` sur les cellules `<td>`.

Il s’agit ensuite de renseigner l’attribut `headers` avec les identifiants des cellules d’entêtes associées. Si plusieurs entêtes sont associés à une cellule de données, les identifiants doivent être séparés par des espaces dans `headers`.

```
<table>
   <caption>Comparatif du chiffre d'affaires des entreprises Dupond et Dupont
en France et dans le monde</caption>
   <tr>
       <th id="entete-1">En millions d'euros</th>
       <th id="entete-2">En France</th>
       <th id="entete-3">Dans le monde</th>
   </tr>
   <tr>
       <th id="entete-4">Dupond</th>
       <td headers="entete-4 entete-2 entete-1">
           50,7
       </td>
       <td headers="entete-4 entete-3 entete-1">
           139,3
       </td>
   </tr>
   <tr>
       <th id="entete-5">Dupont</th>
       <td headers="entete-5 entete-2 entete-1">
           27,1
       </td>
       <td headers="entete-5 entete-3 entete-1">
           476,0
       </td>
   </tr>
</table>
```

Comparatif du chiffre d’affaires des entreprises Dupond et Dupont en France et dans le monde		
En millions d’euros	**En France**	**Dans le monde**
Dupond	50,7	139,3
Dupont	27,1	476,0

Attention

L’attribut `headers` ne doit pas être utilisé en combinaison de l’attribut `scope`.

Remarque

Une bonne pratique d’accessibilité consiste à respecter un ordre logique lorsque les valeurs des attributs id des cellules d’entêtes associées à une cellule de données sont intégrées dans l’attribut `headers` de cette dernière.

En effet, un lecteur d’écran (synthèse vocale et/ou plage braille) annoncera les entêtes dans cet ordre.

9.5. Intégrer role="presentation" dans chaque balise <table> de mise en forme

Signaler les tableaux de mise en forme en intégrant `role="presentation"` dans la balise `<table>`.

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

9.6. Ne pas utiliser de balises ou d'attributs propres aux tableaux de données dans les tableaux de mise en forme

Un tableau de mise en forme ne doit pas posséder de balises ou d'attributs propres aux tableaux de données.

C'est-à-dire :

- Que les balises `<caption>`, `<th>`, `<thead>` et `<tfoot>` ne doivent pas être utilisées dans les tableaux de mise en forme.
- Que les attributs `scope`, `headers`, `axis` et `colgroup` ne doivent pas être utilisés dans les tableaux de mise en forme.

9.7. Veiller à l'ordre de lecture des tableaux de mise en forme

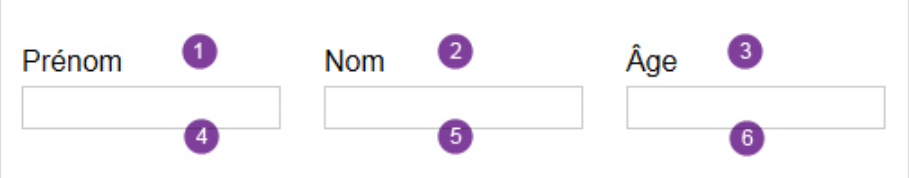
Lorsqu'une personne utilise un lecteur d'écran (synthèse vocale et/ou plage braille) pour accéder à un tableau de mise en forme, son contenu est restitué de manière linéaire. C'est-à-dire que le contenu du tableau est lu cellule après cellule, de gauche à droite et ligne après ligne.

Il faut donc veiller à ce que l'ordre de lecture reste cohérent dans chaque tableau de mise en forme.

Par exemple, si le code source qui suit est utilisé, l'ordre de lecture est :

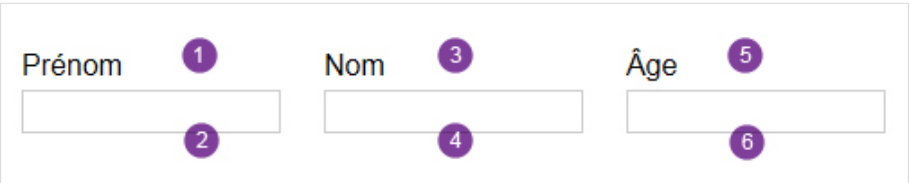
1. Prénom.
2. Nom.
3. Âge.
4. Champ « Prénom ».
5. Champ « Nom ».
6. Champ « Âge ».

```
<table>
 <tr>
  <td><label for="prenom">Prénom</label></td>
  <td><label for="nom">Nom</label></td>
  <td><label for="age">Âge</label></td>
 </tr>
 <tr>
  <td><input type="text" name="prenom" id="prenom" /></td>
  <td><input type="text" name="nom" id="nom" /></td>
  <td><input type="text" name="age" id="age" /></td>
 </tr>
</table>
```



Pour obtenir un ordre de lecture cohérent tout en conservant le tableau de mise en forme, opter pour :

```
<table>
 <tr>
  <td>
   <label for="prenom">Prénom</label>
   <input type="text" name="prenom" id="prenom" />
  </td>
  <td>
   <label for="nom">Nom</label>
   <input type="text" name="nom" id="nom" />
  </td>
  <td>
   <label for="age">Âge</label>
   <input type="text" name="age" id="age" />
  </td>
 </tr>
</table>
```



Attention

Afin de faciliter la lecture par les aides techniques et de garantir l'ordre de lecture, il est fortement recommandé de limiter l'imbrication des tableaux de mise en forme.

10. Usage des CSS

10.1. Utiliser CSS pour mettre en forme les textes

Exemple d'une police de caractère intégrée sous forme de texte.
(Technique utilisée : @font-face CSS3.)

Remarque
Une attention particulière a dû être portée sur la faisabilité de ce point dès la phase de conception graphique.

Si cela n'est pas le cas, se référer à la recommandation « 2.3. Veiller à ce que les polices puissent être intégrées sous forme de texte » de la notice d'accessibilité pour la conception fonctionnelle et graphique et entrer en contact avec la personne en charge de la réalisation des maquettes.

10.2. Utiliser uniquement des tailles relatives (em, %, small, etc.) pour les polices de caractères

Pour définir les tailles des polices de caractères, utiliser uniquement des unités relatives pour la propriété CSS `font-size` comme `em`, `%`, `rem` ou les mots-clés (`x-small`, `small`, etc.).

Ne pas utiliser d'unités absolues comme `pt`, `cm`, etc.

Attention
L'utilisation du pixel (`px`) comme unité de taille de police est interdite dans les référentiels AccessiWeb et RGAA.

Pour garantir la conformité avec ceux-ci, il faudra utiliser une autre unité de taille relative pour la propriété `font-size`.

Remarque
Les CSS d'impression (type de média `print`) ne sont pas concernés par cette recommandation.

10.3. Garantir la lisibilité des contenus même lorsque la taille du texte est doublée

Garantir la lisibilité des contenus même lorsque l'utilisateur double la taille du texte dans son navigateur. Éviter ainsi le chevauchement de texte, la disparition de texte de l'écran, etc.

Pour garantir au maximum le bon respect de cette recommandation, il est recommandé de :

- Ne pas utiliser d'unités (`px`, `pt`, `%`, `em`, etc.) avec la propriété CSS `line-height`.
- Veiller à l'utilisation du positionnement en absolu (déclaration CSS `position: absolute;`). Bien que ce positionnement soit compatible avec l'accessibilité, dans certains cas de figure, il peut entraîner des superpositions de contenu.
- Ne pas définir de hauteur fixe (propriété CSS `height`) sur les éléments susceptibles d'accueillir du contenu textuel, notamment les champs de formulaires.

Astuce

Pour tester cette recommandation :

- Avec Firefox, dans le menu « Affichage » puis « Zoom », cocher « Zoom texte seulement », puis doubler la taille du texte en faisant 6 fois le raccourci clavier `CTRL` et `+` (`CTRL` et `0` permet de revenir à la taille par défaut).
- Avec Internet Explorer, dans le menu « Affichage » puis « Taille du texte », choisir « La plus grande ».

Mobile

Afin de laisser la possibilité aux utilisateurs naviguant avec un périphérique mobile (smartphone/tablette) de zoomer, les déclarations suivantes ne doivent pas être utilisées :

```
<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" />
```

10.4. Garantir la lisibilité des contenus lorsque les images ne sont pas affichées

Lorsque les images ne sont pas affichées, les contenus de la page doivent rester visibles et lisibles. Aucune information ne doit être perdue et le contraste entre la couleur du texte et la couleur d'arrière-plan doit être suffisamment élevé.

C'est-à-dire que tous les contenus doivent pouvoir être lus :

- Même lorsque les images intégrées en CSS ne sont pas chargées dans la page.

- Même lorsque les images intégrées en HTML sont remplacées par le contenu de leur attribut `alt`.

Astuce

Chaque fois qu'un texte est superposé à une image d'arrière-plan, veiller à mettre en place une couleur de remplacement qui garantira la lisibilité du texte en l'absence de cette dernière.

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

Cette couleur de remplacement peut être héritée d'un élément parent.

Attention

Une attention particulière doit être portée sur la lisibilité des alternatives textuelles des images intégrées dans le code HTML lorsque les images ne sont pas affichées.

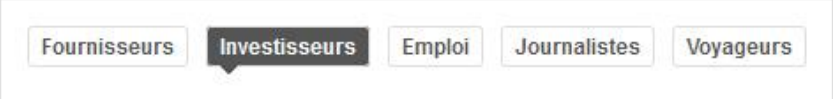
10.5. Garantir la compréhension de l'information même lorsque CSS est désactivé

Veiller à ce que l'information reste disponible et compréhensible même lorsque CSS est désactivé, notamment lorsque des couleurs, des tailles, des formes ou encore des positions sont vecteurs d'informations.

Veiller également à ne pas générer de contenus informatifs en CSS.

Exemples

Position courante dans les menus (exemple 1)



Dans cet exemple, pour le lien « Investisseurs », les couleurs de texte et de fond différentes ainsi qu'une flèche tournée vers le bas indiquent qu'il s'agit de la rubrique courante.

Ci-dessous, deux solutions pour ne pas perdre cette information sans CSS.

Via du texte caché

```
<a href="…">
   Investisseurs
   <span class="hors-ecran">(rubrique courante)</span>
</a>
```

```
.hors-ecran {
   position: absolute;
   left: -99999px;
}
```

Via l'attribut `title`

```
<a href="…" title="Investisseurs (rubrique courante)">
   <strong>Investisseurs</strong>
</a>
```

Astuce

En complément du `title`, une bonne pratique consiste à englober l'intitulé d'une balise `<strong>`.

Position courante dans les menus (exemple 2)

Dans cet exemple, pour le lien « Organigramme », une couleur de fond différente ainsi qu'une mise en gras et une encoche indiquent qu'il s'agit de la page courante.

Ici, comme il s'agit de la page courante, une solution efficace pour véhiculer cette information sans CSS consiste à ne pas englober cet intitulé d'une balise `<a>`.

```
<ul>
   <li><a href="…">Missions</a></li>
   <li><a href="…">Budget</a></li>
   <li><strong>Organigramme</strong></strong>
   <li><a href="…">Instances</a></li>
   <li><a href="…">Dates clés</a></li>
</ul>
```

Contenus générés en CSS (exemple 3)

Les contenus informatifs, nécessaires à la compréhension, ne doivent pas être générés en CSS.

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

Dans ce premier exemple, l'indication du format des fichiers PDF en téléchargement est ajouté en CSS. Ce qui est incorrect.

```
a[target=_blank]::after{
   content: ' (nouvelle fenêtre)';
}
```

Dans ce second exemple, la mention « (nouvelle fenêtre) » est ajoutée en CSS. Ce qui est également incorrect.

11. Navigation au clavier

11.1. Veiller à ce que l'ordre de tabulation suive la logique de l'ordre de lecture

L'ordre de tabulation doit suivre la logique de l'ordre de lecture visuel de la page.

C'est-à-dire que lorsqu'un élément interactif en précède immédiatement un autre lors du parcours visuel de la page, le focus doit continuer sur le second élément immédiatement après avoir quitté le premier.

Attention

Ne pas utiliser l'attribut `tabindex` avec une valeur supérieure à 0 au risque de perturber la logique de l'ordre de tabulation dans la page.

Remarque

Dans certains composants d'interface riche, comme par exemple les systèmes d'onglets, la navigation d'un élément interactif à un autre se fait avec les flèches directionnelles plutôt qu'avec la touche `Tab`.

11.2. Mettre en place des liens d'évitement

Un lien d'évitement du type « Aller au contenu » doit systématiquement être présent sur chaque page afin de faciliter la navigation au clavier.

Ce lien doit être le premier élément interactif dans le code HTML.

Il s'agit d'un lien interne qui doit permettre un accès direct au contenu principal de la page.

```
<a class="evitement" href="#contenu">Aller au contenu</a>
[…]
<main role="main" id="contenu" tabindex="-1">[…]</main>
```

Remarque

L'intégration de `tabindex="-1"` dans la balise de destination permet de rendre ce lien-ancre fonctionnel sous Internet Explorer et Chrome.

Attention

Bien qu'il soit recommandé d'afficher ce lien, celui-ci peut être masqué par défaut. En revanche, il doit dans tous les cas être rendu visible à la prise de focus au clavier.

Par conséquent, le lien d'évitement ne doit jamais être masqué à l'aide des propriétés CSS `display: none;` et/ou `visibility: hidden;` sous peine de le rendre totalement inatteignable au clavier.

Privilégier une autre solution, par exemple l'utilisation des codes suivants :

```
a.evitement {
   position: absolute;
   left: -99999px;
}

a.evitement:focus {
   position: static;
}
```

Remarque

Dans certaines situations, de nombreuses tabulations sont nécessaires pour accéder aux menus principal/secondaire et/ou au moteur de recherche depuis le sommet de la page.

Dans ce cas-là, mettre en place une liste de plusieurs liens d'évitement. Comme par exemple :

```
<ul id="evitement">
   <li>
      <a href="#contenu">Aller au contenu</a>
   </li>
   <li>
      <a href="#menu">Aller au menu</a>
   </li>
   <li>
      <a href="#recherche">Aller à la recherche</a>
   </li>
</ul>
```

11.3. Garantir la visibilité de la prise de focus au clavier

Afin de permettre aux utilisateurs qui naviguent au clavier de se situer dans la page, chaque élément interactif (liens, boutons, champs de formulaires, etc.) doit être visuellement mis en avant lors de la prise de focus.

Astuce

Une bonne pratique d'accessibilité consiste à doubler systématiquement chaque règle `:hover` par une règle `:focus` dans la CSS.

Comme par exemple :

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

12. Scripts

12.1. Permettre le contrôle des scripts à la fois à la souris et au clavier

Chaque fois qu'un script est contrôlable à la souris, il doit être possible de contrôler ce dernier également au clavier (et inversement).

C'est-à-dire que chaque fois que la souris permet de contrôler un script depuis un élément (un lien ou un bouton, par exemple), celui-ci doit :

- Être atteignable au clavier.
- Permettre le contrôle du script au clavier une fois le focus placé sur l'élément.

Attention
Les raccourcis clavier varient d'un système à l'autre. Dès lors, chaque fois que cela est possible, éviter les interactions liées à l'action sur des touches spécifiques du clavier.

Par exemple, pour déclencher une action à la perte de focus clavier sur un élément, privilégier l'écoute de l'événement générique onblur plutôt que celle de l'état des touches `Tab` et `Maj` + `Tab`.

Mobile
Veiller à ce que les interactions tactiles fonctionnent également à la souris et au clavier.

13. Recommandations additionnelles

Dans une optique de pragmatisme, plusieurs critères d'accessibilité présents dans les référentiels d'accessibilité n'ont pas été conservés dans les recommandations de cette notice.

Il s'agit de critères rarement applicables dans un projet web ou de critères moins prioritaires. Le détail de ces recommandations est disponible à l'adresse suivante : www.accede-web.com/notices/html-css-javascript/13-recommandations-additionnelles/.

Pour l'atteinte du niveau A (WCAG 2.0, RGAA 3, AccessiWeb HTML5/ARIA)

- Renseigner l'attribut `alt` de chaque image mappée et de ses balises `<area />`.
- Baliser les blocs de citations avec `<blockquote>`.
- Baliser les citations en ligne avec `<q>`.
- Renseigner l'attribut `title` sur chaque `<iframe>`.
- Renseigner l'attribut `title` sur chaque `<frame>`.
- Regrouper les options de même nature avec `<optgroup>` dans les `<select>`.
- Indiquer dans l'alternative des CAPTCHA graphiques où trouver la version non graphique du CAPTCHA.
- Indiquer la langue de chaque document en téléchargement rédigé dans une langue étrangère.
- Doubler les images dotées d'un attribut `ismap` avec des liens alternatifs.
- Renseigner le sens de lecture principal de la page avec l'attribut `dir="ltr"` (gauche à droite) ou `dir="rtl"` (droite à gauche) sur la balise `<html>`.
- Utiliser l'attribut `dir` pour signaler les changements de sens de lecture dans le corps de la page.
- Identifier les principales zones de la page avec l'attribut `id` ou une ancre nommée.
- Prévoir une alternative à chaque contenu multimédia (`<video>`, `<audio>`, `<object>`, etc.).
- Ne pas utiliser le paramètre `wmode` avec les valeurs transparent ou opaque.
- Ne pas intégrer de contenu textuel dans les balises `<canvas>` décoratives.
- Intégrer une alternative textuelle pertinente dans les balises `<canvas>` informatives ou servant de liens.

Pour l'atteinte du niveau AA (WCAG 2.0, RGAA 3, AccessiWeb HTML5/ARIA)

Aucune recommandation additionnelle au niveau AA (WCAG 2.0, RGAA 3, AccessiWeb HTML5/ARIA).

Pour l’atteinte du niveau AAA (WCAG 2.0, RGAA 3, AccessiWeb HTML5/ARIA)

- Prévoir des intitulés de liens explicites hors contexte et sans l’utilisation de l’attribut `title`.
- Baliser les définitions avec `<dfn>`.
- Baliser les abréviations avec `<abbr>`.
- Limiter la largeur des textes à 80 caractères, quelle que soit la taille du texte.
- Prévoir un interligne d’au moins 1,5 fois la taille du texte, quelle que soit la taille du texte.
- Prévoir une marge entre les paragraphes d’au moins 1,5 fois la valeur de l’interligne, quelle que soit la taille du texte.
- Garantir l’absence de barre de défilement horizontale même lorsque la taille du texte est doublée par l’utilisateur.
- Garantir des longueurs de lignes inférieures à 80 caractères lorsque l’utilisateur réduit la largeur de la fenêtre du navigateur.