ATALAN
Accessibilité numérique
Sensibilisation au handicap
www.atalan.fr

AcceDe Web
La démarche accessibilité
www.accede.info

Notice d'accessibilité des principaux composants d'interface riche

Date	Version	État / commentaires
15 mars 2016	2.1	Cette version prend en compte WCAG 2.0 et RGAA 3.0.
23 septembre 2020	4.0	Mises à jour majeures pour être conforme avec le RGAA 4.

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

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 lors de la conception fonctionnelle et du développement des principaux composants d'interface riche.

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

- Notice d'accessibilité fonctionnelle et graphique.
- Notice d'accessibilité HTML et CSS.
- **Notice d'accessibilité des principaux composants d'interface riche (présente notice).**
- 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 et l'intégration ou le développement de composants d'interfaces riches. 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 de sélection ou de développement de composants d'interfaces riches.

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/interface-riche/.

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

Accordéons

Principe

Les accordéons sont des composants dynamiques qui permettent d'optimiser l'affichage d'un contenu dans un espace réduit grâce à un système de « plier/déplier » appliqué sur un groupe de panneaux.

Ils sont contrôlables via une action sur un bouton qui surplombe le contenu de chaque panneau.

Cette fiche s'appuie sur le motif de conception « Accordion » détaillé dans les WAI-ARIA 1.1 Authoring Practices du W3C.

Socle HTML

```
<h2>
  <button aria-expanded="false" aria-controls="accordion-panel-1">Entête du panneau 1</button>
</h2>
<div id="accordion-panel-1">
  [Contenu du panneau 1 (replié)]
</div>
<h2>
  <button aria-expanded="true" aria-controls="accordion-panel-2">Entête du panneau 2</button>
</h2>
<div id="accordion-panel-2">
  [Contenu du panneau 2 (déplié)]
</div>
<h2>
  <button aria-expanded="false" aria-controls="accordion-panel-3">Entête du panneau 3</button>
</h2>
<div id="accordion-panel-3">
  [Contenu du panneau 3 (replié)]
</div>
```

Rôles, états et propriétés ARIA

- Chaque entête de panneau doit être balisé avec un `<button>`.
- Chaque bouton d'entête de panneau doit être entouré d'une balise de titre (`<h1>` à `<h6>`) dont le niveau dépend du contexte dans lequel l'accordéon est intégré.

- L'attribut `aria-expanded` doit être appliqué sur chaque bouton d'entête de panneau. Sa valeur doit être renseignée dynamiquement en fonction de l'état du panneau associé :
  - `aria-expanded="true"` lorsque le panneau associé est déplié.
  - `aria-expanded="false"` lorsque le panneau associé est replié.
- Chaque bouton d'entête de panneau doit être rattaché à son panneau associé via l'attribut `aria-controls` :
  - Chaque panneau doit posséder un attribut `id` renseigné avec une valeur unique.
  - Chaque bouton d'entête de panneau doit posséder un attribut `aria-controls` renseigné avec la valeur de l'attribut `id` du panneau associé.

Interactions au clavier

Entrée ou Espace

- Si le focus clavier est positionné sur le bouton d'entête d'un panneau replié, déplie le panneau associé. Si l'accordéon n'autorise le déploiement que d'un seul panneau à la fois, et si un autre panneau est déjà déplié, replie ce panneau.
- Si le focus clavier est positionné sur le bouton d'entête d'un panneau déplié, replie le panneau associé si l'accordéon autorise le repliement de ce panneau. Certains accordéons exigent qu'un seul et unique panneau soit déplié à tout instant ; dans ce cas de figure, le panneau déplié ne peut pas être replié via une action sur son bouton d'entête associé.

Remarque

Dans le socle HTML proposé sur cette fiche, des titres de niveau 2 (`<h2>`) sont utilisés pour baliser les entêtes des panneaux d'accordéon. Le niveau de ces titres est éventuellement à adapter en fonction du contexte d'intégration de ce composant : il est important de veiller à maintenir une hiérarchie des titres logique dans la page.

Ainsi, si un `<h2>` surplombe le système d'accordéon, chaque entête de panneau devient enfant de ce titre de niveau 2 et devra donc être balisé avec `<h3>`.

Composants

Ces composants « Accordéons » sont proposés ici car leur niveau d'accessibilité est jugé bon ou très bon.

Toutefois, avant de les utiliser dans votre projet, il est important de vérifier le respect des spécifications présentées ci-avant. Certains pouvant nécessiter quelques ajustements.

Fenêtres modales

Principe

Les fenêtres modales sont des fenêtres qui apparaissent directement à l'intérieur de la fenêtre courante du navigateur, *au-dessus* de la page web qui les appelle.

Il s'agit de fenêtres modales, c'est-à-dire de fenêtres qui prennent le contrôle de la page courante tant qu'elles sont affichées à l'écran.

Cette fiche s'appuie sur le motif de conception « Dialog (modal) » détaillé dans les WAI-ARIA 1.1 Authoring Practices du W3C.

Socle HTML

Le socle HTML d'une fenêtre modale est différent selon que cette dernière possède un titre affiché à l'écran ou non.

Fenêtre modale *avec* un titre affiché à l'écran

```
<div role="dialog" aria-modal="true" aria-labelledby="modal-heading">
  <button>Fermer</button>
  <h1 id="modal-heading">[Titre de la modale]</h1>
  [Contenu de la modale]
</div>
```

Fenêtre modale *sans* titre affiché à l'écran

```
<div role="dialog" aria-modal="true" aria-label="[Titre de la modale]">
   <button>Fermer</button>
   [Contenu de la modale]
 </div>
```

Rôles, états et propriétés ARIA

- `role="dialog"` doit être appliqué sur le conteneur de la fenêtre modale.
- `aria-modal="true"` doit être appliqué sur le conteneur de la fenêtre modale.
- Si le titre de la fenêtre modale est affiché à l'écran, il doit être rattaché à la fenêtre modale via l'attribut `aria-labelledby` :
  - Le titre de la fenêtre modale doit posséder un attribut `id` renseigné avec une valeur unique.
  - Le conteneur de la fenêtre modale doit posséder un attribut `aria-labelledby` renseigné avec la valeur de l'attribut `id` du titre de la fenêtre modale.
- Si le titre de la fenêtre modale n'est pas affiché à l'écran, `aria-label` doit être appliqué et renseigné sur le conteneur de la fenêtre modale.

Interactions au clavier

Tab

Lorsque la fenêtre modale est affichée, déplace successivement le focus clavier vers chacun des éléments interactifs contenus dans la fenêtre modale. Si le focus clavier est positionné au niveau du dernier élément interactif contenu dans la boîte de dialogue modale au moment où la touche est pressée, le focus clavier est déplacé au niveau du premier élément interactif contenu dans la fenêtre modale.

Maj + Tab

Même comportement qu'avec la touche **Tab**, mais cette fois dans l'ordre inverse de lecture. Si le focus clavier est positionné au niveau du premier élément interactif contenu dans la fenêtre modale au moment où la combinaison de touches est pressée, le focus clavier est déplacé au niveau du dernier élément interactif contenu dans la fenêtre modale.

Échap

Lorsque la boîte de dialogue modale est affichée, ferme la fenêtre modale et déplace le focus clavier sur l'élément interactif qui a déclenché l'ouverture de la fenêtre modale.

Comportements attendus

Lorsque la fenêtre modale est affichée à l'écran

- Le focus clavier est positionné dynamiquement sur le premier élément interactif contenu dans la fenêtre modale.
- Le focus clavier doit être bloqué à l'intérieur de la fenêtre modale et il ne doit pas être possible de tabuler sur le reste de la page, en-dessous de la fenêtre modale.
- Il est possible de fermer la fenêtre modale avec la touche **Échap**.

Lorsque la fenêtre modale est fermée

- Le focus clavier doit être replacé au niveau de l'élément qui a déclenché l'ouverture de la fenêtre modale.
- Dans l'idéal, la fenêtre modale est supprimée du DOM. Toutefois, si la fenêtre modale reste présente dans le code source, `display: none` ou `visibility: hidden` doivent être appliqués sur son conteneur.

Composants

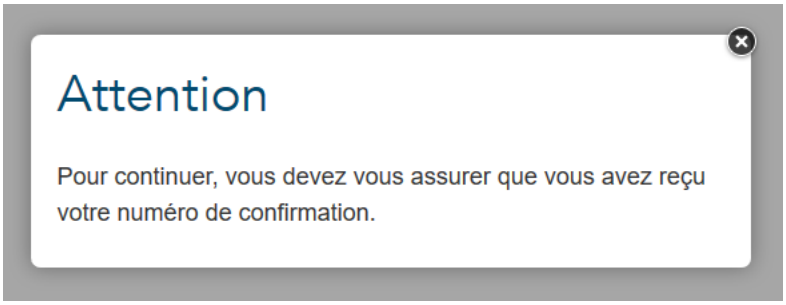
Ces composants « Fenêtres modales » sont proposés ici car leur niveau d'accessibilité est jugé bon ou très bon.

Toutefois, avant de les utiliser dans votre projet, il est important de vérifier le respect des spécifications présentées ci-avant. Certains pouvant nécessiter quelques ajustements.

Fenêtres d’alerte modales

Principe

Les boîtes d’alerte modales sont un cas particulier de fenêtres modales.



Elles renvoient une alerte concise ou demandent une confirmation rapide et sont adaptées lorsque :

- Le message ne dépasse pas une phrase.
- La ponctuation n’est pas essentielle à la compréhension du message. Les boîtes d’alerte modales sont donc par exemple inadaptées pour annoncer qu’une syntaxe précise du type « JJ/MM/AAAA » est attendue pour un champ de date.
- Le message ne contient pas d’informations que la personne pourrait avoir besoin de réutiliser, comme un numéro de téléphone.
- Le message ne contient pas d’éléments interactifs, comme un lien vers une ressource.

Cette fiche s’appuie sur le motif de conception « Alert Dialog » détaillé dans les WAI-ARIA 1.1 Authoring Practices du W3C.

Socle HTML

Le socle HTML d’une fenêtre d’alerte modale est différent selon que cette dernière possède un titre affiché à l’écran ou non.

Fenêtre d’alerte modale *avec* un titre affiché à l’écran :

```
<div role="alertdialog" aria-modal="true" aria-labelledby="modal-heading"
aria-describedby="modal-content">
  <button>Fermer</button>
  <h1 id="modal-heading">[Titre de la modale]</h1>
  <p id="modal-content">[Contenu de la modale]</p>
</div>
```

Fenêtre d'alerte modale *sans* titre affiché à l'écran :

```
<div role="alertdialog" aria-modal="true" aria-label="[Titre de la modale]"
aria-describedby="modal-content">
   <button>Fermer</button>
   <p id="modal-content">[Contenu de la modale]</p>
 </div>
```

Rôles, états et propriétés ARIA

- `role="alertdialog"` doit être appliqué sur le conteneur de la fenêtre d'alerte modale.
- `aria-modal="true"` doit être appliqué sur le conteneur de la fenêtre d'alerte modale.
- Si le titre de la fenêtre d'alerte modale est affiché à l'écran, il doit être rattaché à la fenêtre d'alerte modale via l'attribut `aria-labelledby` :
  - Le titre de la fenêtre d'alerte modale doit posséder un attribut `id` renseigné avec une valeur unique.
  - Le conteneur de la fenêtre d'alerte modale doit posséder un attribut `aria-labelledby` renseigné avec la valeur de l'attribut `id` du titre de la fenêtre d'alerte modale.
- Si le titre de la fenêtre d'alerte modale n'est pas affiché à l'écran, `aria-label` doit être appliqué et renseigné sur le conteneur de la fenêtre d'alerte modale.
- Le message doit être rattaché à la boîte d'alerte modale via l'attribut `aria-describedby` :
  - Le message doit posséder un attribut `id` renseigné avec une valeur unique.
  - Le conteneur de la boîte d'alerte modale doit posséder un attribut `aria-describedby` renseigné avec la valeur de l'attribut `id` du message.

Interaction au clavier

Tab

Lorsque la fenêtre d'alerte modale est affichée, déplace successivement le focus clavier vers chacun des éléments interactifs contenus dans la fenêtre d'alerte modale. Si le focus clavier est positionné au niveau du dernier élément interactif contenu dans la boîte de dialogue modale au moment où la touche est pressée, le focus clavier est déplacé au niveau du premier élément interactif contenu dans la fenêtre d'alerte modale.

Maj + **Tab**

Même comportement qu'avec la touche **Tab**, mais cette fois dans l'ordre inverse de lecture. Si le focus clavier est positionné au niveau du premier élément interactif contenu dans la fenêtre d'alerte modale au moment où la combinaison de touches est pressée, le focus clavier est déplacé au niveau du dernier élément interactif contenu dans la fenêtre d'alerte modale.

Échap

Lorsque la boîte de dialogue modale est affichée, ferme la fenêtre d'alerte modale, et déplace le focus clavier sur l'élément interactif qui a déclenché l'ouverture de la fenêtre d'alerte modale.

Comportements attendus

Lorsque la fenêtre d'alerte modale est affichée à l'écran

- Le focus clavier est positionné dynamiquement sur le premier élément interactif contenu dans la fenêtre d'alerte modale.
- Le focus clavier doit être bloqué à l'intérieur de la fenêtre d'alerte modale et il ne doit pas être possible de tabuler sur le reste de la page, en-dessous de la fenêtre d'alerte modale.
- Il est possible de fermer la fenêtre d'alerte modale avec la touche **Échap**

Lorsque la fenêtre d'alerte modale est fermée

- Le focus clavier doit être replacé au niveau de l'élément qui a déclenché l'ouverture de la fenêtre d'alerte modale.
- Dans l'idéal, la fenêtre d'alerte modale est supprimée du DOM. Toutefois, si la fenêtre d'alerte modale reste présente dans le code source, `display: none` ou `visibility: hidden` doivent être appliqués sur son conteneur.

Composants

Ces composants « Fenêtres d'alerte modales » sont proposés ici car leur niveau d'accessibilité est jugé bon ou très bon.

Toutefois, avant de les utiliser dans votre projet, il est important de vérifier le respect des spécifications présentées ci-avant. Certains pouvant nécessiter quelques ajustements.

Boutons « Afficher plus »

Principe

Les boutons « Afficher plus » sont des composants dynamiques qui se présentent sous la forme d'un bouton permettant l'affichage d'un contenu complémentaire juste avant ce dernier, à chaque fois que le bouton est activé, et tant que du contenu complémentaire reste disponible.

Les boutons « Afficher plus » disparaissent lorsque la totalité du contenu complémentaire a été affiché dans la page.

Socle HTML

Avant activation du bouton

```
<div>[Contenu affiché par défaut]</div>
<button>[Intitulé du bouton "Afficher plus"]<button>
```

Après activation du bouton

```
div>[Contenu affiché par défaut]</div>
<div tabindex="-1">[Contenu complémentaire, nouvellement affiché]</div>
<button>[Intitulé du bouton "Afficher plus"]<button>
```

Rôles, états et propriétés ARIA

Suite à l'activation du bouton, `tabindex="-1"` doit être appliqué sur le conteneur du contenu nouvellement affiché.

Interactions au clavier

Entrée et **Espace**

Lorsque le focus clavier est positionné sur le bouton, affiche incrémentalement du contenu complémentaire, tant que celui-ci est disponible.

Comportements attendus

- Lorsque le focus clavier est positionné au niveau du bouton, il est possible d'afficher du contenu complémentaire avec les touches **Espace** et **Entrée**. Pour cela, passer par l'écoute de l'événement `click`.
- Lorsque le bouton est activé, le focus clavier est positionné dynamiquement au niveau du conteneur du contenu complémentaire nouvellement affiché.
- Lorsqu'il ne reste plus de contenu complémentaire à afficher, le bouton est supprimé.

Remarque

Dans le cas particulier où le bouton ne permet d'afficher que des éléments interactifs, il est possible de se passer de l'attribut `tabindex="-1"`, et de se contenter de positionner le focus sur le premier élément interactif nouvellement apparu, une fois le bouton activé.

Ainsi, par exemple, dans le cas d'un bouton « Afficher plus d'actualités » qui déclenche l'affichage de liens vers davantage d'actualités :

```
<ul>
  <li><a href="…">Actualité 1</a></li>
  <li><a href="…">Actualité 2</a></li>
  <li><a href="…">Actualité 3</a></li>
</ul>
<button>Afficher plus d'actualités</button>
```

Une fois le bouton activé, le focus clavier doit être positionné au niveau du lien « Actualité 4 » nouvellement apparu.

L'emploi de `tabindex="-1"` n'est pas nécessaire ici, car l'élément `<a>` est capable de recevoir le focus clavier par défaut.

```
<ul>
  <li><a href="…">Actualité 1</a></li>
  <li><a href="…">Actualité 2</a></li>
  <li><a href="…">Actualité 3</a></li>
  <li><a href="…">Actualité 4</a></li>
  <li><a href="…">Actualité 5</a></li>
  <li><a href="…">Actualité 6</a></li>
</ul>
```

Boutons radio personnalisés en ARIA

Principe

Les boutons radio sont des éléments de formulaires qui permettent de sélectionner une option unique parmi un groupe d'options proposées.

Les boutons radio sont dits « personnalisés en ARIA » lorsqu'ils ne sont pas construits sur la base du code HTML standard prévu pour ces éléments par la spécification : `<input type="radio" />`.

Le suivi des recommandations ci-dessous permet de reproduire le comportement par défaut des boutons radio HTML standards, **dans les cas où ces derniers ne peuvent être utilisés**.

Cette fiche s'appuie sur le motif de conception « Radio Group » détaillé dans les WAI-ARIA 1.1 Authoring Practices du W3C.

Socle HTML

```
<p id="question">Question</p>
<ul aria-labelledby="question" role="radiogroup">
  <li role="radio" aria-checked="false" tabindex="-1">
    <img aria-hidden="true" src="radio.svg" alt="Non sélectionné : " />
    Choix 1
  </li>
  <li role="radio" aria-checked="true" tabindex="0">
    <img aria-hidden="true" src="radio-checked.svg" alt="Sélectionné : " />
    Choix 2
  </li>
  <li role="radio" aria-checked="false" tabindex="-1">
    <img aria-hidden="true" src="radio.svg" alt="Non sélectionné : " />
    Choix 3
  </li>
</ul>
```

Rôles, états et propriétés ARIA

- `role="radiogroup"` doit être appliqué sur le groupe de boutons radio.
- `role="radio"` doit être appliqué sur le conteneur de chaque bouton radio.
- L'attribut `tabindex` doit être appliqué par défaut sur le conteneur de chaque bouton radio. Sa valeur doit être renseignée dynamiquement en fonction de l'état des boutons radio :
  - Si aucun bouton radio n'est sélectionné : `tabindex="0"` sur le premier et le dernier bouton radio du groupe, `tabindex="-1"` sur les autres boutons radio.
  - Si un bouton radio est sélectionné : `tabindex="0"` sur le bouton radio sélectionné, `tabindex="-1"` sur les autres boutons radio.

- `aria-hidden="true"` doit être appliqué sur chaque image simulant un bouton radio.
- L'attribut `aria-checked` doit être appliqué sur le conteneur de chaque bouton radio. Sa valeur doit être renseignée dynamiquement en fonction de l'état du bouton radio associé :
  - `aria-checked="true"` lorsque le bouton radio est sélectionné.
  - `aria-checked="false"` lorsque le bouton radio n'est pas sélectionné.
- Le groupe de boutons radio doit être rattaché à l'étiquette du groupe via l'attribut `aria-labelledby` :
  - L'étiquette du groupe doit posséder un attribut `id` renseigné avec une valeur unique.
  - Le groupe de boutons radio doit posséder un attribut `aria-labelledby` renseigné avec la valeur de l'attribut `id` de l'étiquette du groupe.

Interactions au clavier

Les interactions au clavier sont les mêmes que pour des boutons radio HTML classiques. À la seule exception que le focus clavier est ici positionné sur le conteneur du bouton radio et non uniquement sur le bouton radio.

Tab et **Maj** + **Tab**

Lorsque l'on accède en tabulant à un groupe de boutons radio, le focus est positionné au niveau de l'éventuel bouton radio sélectionné dans le groupe. Lorsque le focus est positionné sur un bouton radio sélectionné, la prochaine tabulation permet de quitter le groupe de boutons radio.

Dans le cas où aucun bouton radio n'est sélectionné au moment où l'on accède au groupe de boutons radio au clavier, le focus clavier est positionné :

- Au niveau du premier bouton radio du groupe si la touche **Tab** a été pressée.
- Au niveau du dernier bouton radio du groupe si les touches **Maj** + **Tab** ont été pressées.

Flèche haut et **Flèche gauche**

Lorsque le focus est positionné sur l'un des boutons radio, déplace le focus clavier vers le précédent bouton radio dans le groupe et sélectionne ce bouton. Si le focus clavier est positionné au niveau du premier bouton radio du groupe au moment où la touche est pressée, le focus clavier est déplacé au niveau du dernier bouton radio du groupe et ce bouton est sélectionné.

Flèche bas et **Flèche droite**

Lorsque le focus est positionné sur l'un des boutons radio, déplace le focus clavier vers le bouton radio suivant dans le groupe et sélectionne ce bouton. Si le focus clavier est positionné au niveau du dernier bouton radio du groupe au moment où la touche est pressée, le focus clavier est déplacé au niveau du premier bouton radio du groupe et ce bouton est sélectionné.

Espace

Lorsque le focus clavier est positionné sur un bouton radio, sélectionne le bouton radio et désélectionne l'éventuel autre bouton radio déjà sélectionné dans le groupe.

Remarques

La source de l'image, ainsi que son texte alternatif, doivent être modifiés en fonction de l'état du bouton radio correspondant.

À noter qu'une balise `<img />` est proposée dans l'exemple de code, mais qu'il est également possible de la remplacer par une image vectorielle `<svg>`.

Composants

Ces composants « Boutons radio personnalisés en ARIA » sont proposés ici car leur niveau d'accessibilité est jugé bon ou très bon.

Toutefois, avant de les utiliser dans votre projet, il est important de vérifier le respect des spécifications présentées ci-avant. Certains pouvant nécessiter quelques ajustements.

Boutons radio personnalisés en CSS

Principe

Les boutons radio sont des éléments de formulaires qui permettent de sélectionner une option unique parmi un groupe d'options proposées.

Les boutons radio sont dits « personnalisés en CSS » lorsqu'ils sont construits sur la base du code HTML standard prévu pour ces éléments par la spécification : `<input type="radio" />`, mais que ces boutons radio standards sont masqués à l'écran puis simulés en CSS grâce à des images, des polices d'icônes ou des styles spécifiques.

De nombreuses approches sont envisageables. L'exemple ci-dessous expose comment personnaliser des boutons radio à l'aide de pseudo-éléments CSS et d'images d'arrière-plan, tout en conservant leur caractère accessible.

Socle HTML

```
<fieldset>
  <legend>Question</legend>
  <ul>
   <li>
    <input type="radio" id="reponse-1" name="question" value="reponse-1" />
    <label for="reponse-1">Réponse 1</label>
   </li>
   <li>
    <input type="radio" id="reponse-2" name="question" value="reponse-2" />
    <label for="reponse-2">Réponse 2</label>
   </li>
   [...]
  </ul>
</fieldset>
```

Socle CSS

```
label {
  padding: 0 0 0 2rem;
  position: relative;
}

input[type=radio] {
  position: absolute;
  opacity: 0;
}

input[type=radio] + label::before,
input[type=radio] + label::after {
  content: '';
  position: absolute;
  border-radius: 50%;
}
```

```css
input[type=radio] + label::before {
  left: 0.5rem;
  top: 0.2rem;
  display: inline-block;
  width: 0.8rem;
  height: 0.8rem;
  border: 0.05rem solid black;
  background: white;
}

input[type=radio]:checked + label::after {
  left: 0.7rem;
  top: 0.4rem;
  border: 0.25rem solid black;
}

input[type=radio]:focus + label::before {
  outline: 0.05rem dotted;
}
```

Carrousels

Depuis la rédaction de cette fiche, une mise à jour de la spécification ARIA a eu lieu : passage de ARIA 1.0 (ancienne version) à ARIA 1.1 (nouvelle version).

Cette fiche s'appuie sur les motifs de conception ARIA 1.0 issus du document WAI-ARIA Authoring Practices 1.0, et les recommandations ci-dessous pourront faire l'objet d'ajustements une fois les motifs de conception ARIA 1.1 officialisés.

Pour plus d'informations sur les changements relatifs au passage à ARIA 1.1, consulter les WAI-ARIA Authoring Practices 1.1.

Principe

Les carrousels sont des modules dynamiques qui permettent d'optimiser l'affichage du contenu d'une page dans un espace réduit grâce à un système de navigation contrôlant l'affichage de panneaux défilants, parfois automatiquement.

Ils se présentent généralement sous la forme d'un panneau de contenu entouré de boutons « Précédent » et « Suivant » qui permettent de faire défiler les panneaux du carrousel. Ils sont souvent accompagnés d'un système de pagination.

Dans le cas d'un carrousel à défilement automatique, un système de mise en pause et de relance du défilement est présent.

Ce carrousel animé dispose d'un système de mise en pause du mouvement.

Cette fiche s'appuie sur le motif de conception « Tab Panel » détaillé dans les WAI-ARIA 1.0 Authoring Practices du W3C.

Socle HTML

```
<button><img src="pause.png" alt="Arrêter le défilement du carrousel"
/></button>
<ul role="tablist">
  <li role="tab" tabindex="-1" aria-selected="false" aria-
controls="panneau-1"><img src="puce.png" alt="Panneau 1" /></li>
  <li role="tab" tabindex="0" aria-selected="true" aria-controls="panneau-
2"><img src="puce-active.png" alt="Panneau 2" /></li>
  <li role="tab" tabindex="-1" aria-selected="false" aria-
controls="panneau-3"><img src="puce.png" alt="Panneau 3" /></li>
</ul>
<button><img src="precedent.png" alt="Panneau précédent" /></button>
<div>
  <div role="tabpanel" id="panneau-1" aria-hidden="true">
    [Contenu du premier panneau (masqué)]
    <a href="#" tabindex="-1">Exemple de lien</a>
  </div>
  <div role="tabpanel" id="panneau-2" aria-hidden="false">
    [Contenu du deuxième panneau (affiché, car onglet associé sélectionné)]
    <a href="#">Exemple de lien</a>
  </div>
  <div role="tabpanel" id="panneau-3" aria-hidden="true">
    [Contenu du troisième panneau (masqué)]
    <a href="#" tabindex="-1">Exemple de lien</a>
  </div>
</div>
<button><img src="suivant.png" alt="Panneau suivant" /></button>
```

Rôles, états et propriétés ARIA

- `role="tablist"` doit être appliqué sur l'élément englobant les onglets de pagination.
- `role="tab"` doit être appliqué sur chaque onglet de pagination.
- L'attribut `tabindex` doit être appliqué sur chaque onglet de pagination. Sa valeur doit être renseignée dynamiquement en fonction de l'état de l'onglet de pagination associé :
  - `tabindex="0"` sur l'onglet de pagination sélectionné.
  - `tabindex="-1"` sur les autres onglets de pagination.
- L'attribut `aria-selected` doit être appliqué sur chaque onglet de pagination. Sa valeur doit être renseignée dynamiquement en fonction de l'état de l'onglet de pagination associé :
  - `aria-selected="true"` sur l'onglet de pagination sélectionné.
  - `aria-selected="false"` sur les autres onglets de pagination.

- `role="tabpanel"` doit être appliqué sur chaque panneau.
- L'attribut `aria-hidden` doit être appliqué sur chaque panneau. Sa valeur doit être renseignée dynamiquement en fonction de l'état du panneau associé :
  - `aria-hidden="false"` sur le panneau affiché.
  - `aria-hidden="true"` sur les autres panneaux.
- `tabindex="-1"` doit être appliqué dynamiquement sur chaque élément interactif contenu dans un panneau masqué. L'attribut ne doit pas être présent sur les éléments interactifs contenus dans le panneau affiché.
- Chaque onglet de pagination doit être rattaché à son panneau associé via l'attribut `aria-controls` :
  - Chaque panneau doit posséder un attribut `id` renseigné avec une valeur unique.
  - Chaque onglet de pagination doit posséder un attribut `aria-controls` renseigné avec la valeur de l'attribut `id` du panneau associé.

Interactions au clavier

Tab

Lorsque l'on accède en tabulant au groupe d'onglets de pagination, le focus est positionné au niveau de l'onglet de pagination sélectionné dans le groupe d'onglets de pagination. Lorsque le focus est positionné sur un onglet de pagination, la prochaine tabulation permet de quitter le groupe d'onglets de pagination.

Maj + **Tab**

Même comportement qu'avec la touche **Tab**, mais cette fois dans l'ordre inverse de lecture.

Flèche haut et **Flèche gauche**

Lorsque le focus est positionné sur un onglet de pagination, déplace le focus clavier vers le précédent onglet du groupe d'onglets de pagination et sélectionne cet onglet de pagination. Si le focus clavier est positionné au niveau du premier onglet de pagination du groupe au moment où la touche est pressée, le focus clavier est déplacé au niveau du dernier onglet de pagination du groupe et cet onglet de pagination est sélectionné.

Flèche bas et **Flèche droite**

Lorsque le focus est positionné sur un onglet de pagination, déplace le focus clavier vers l'onglet de pagination suivant dans le groupe et sélectionne cet onglet de pagination. Si le focus clavier est positionné au niveau du dernier onglet de pagination du groupe au moment où la touche est pressée, le focus clavier est déplacé au niveau du premier onglet de pagination du groupe et cet onglet de pagination est sélectionné.

Ctrl + **Flèche haut**

Lorsque le focus est positionné dans un panneau, déplace le focus clavier vers l'onglet de pagination associé à ce panneau.

Ctrl + **Page précédente**

Lorsque le focus est positionné dans un panneau, déplace le focus clavier vers le précédent onglet de pagination du groupe et sélectionne cet onglet de pagination. Si le focus clavier est positionné au niveau du premier panneau du groupe au moment où la combinaison de touches est pressée, le focus clavier est déplacé au niveau du dernier onglet de pagination du groupe et cet onglet de pagination est sélectionné.

Ctrl + **Page suivante**

Lorsque le focus est positionné dans un panneau, déplace le focus clavier vers l'onglet de pagination suivant dans le groupe et sélectionne cet onglet de pagination. Si le focus clavier est positionné au niveau du dernier panneau du groupe au moment où la combinaison de touches est pressée, le focus clavier est déplacé au niveau du premier onglet de pagination du groupe et cet onglet de pagination est sélectionné.

Entrée et **Espace**

Lorsque le focus clavier est positionné sur les boutons de navigation, affiche le panneau précédent ou suivant.

Lorsque le focus clavier est positionné sur le bouton de mise en pause et de relance du défilement, stoppe et relance alternativement le mouvement.

Comportements attendus

- Parmi tous les onglets de pagination, un seul onglet de pagination peut être sélectionné à la fois et seul l'onglet de pagination actif peut recevoir le focus clavier.
- Lorsqu'un onglet de pagination inactif est sélectionné, l'onglet de pagination précédemment sélectionné devient inactif et le focus clavier est positionné au niveau de l'onglet de pagination nouvellement sélectionné.

- Un seul panneau peut être affiché à la fois. Seul le panneau associé à l'onglet de pagination en cours de sélection est affiché. Les autres panneaux sont masqués avec `aria-hidden="true"` et éventuellement `display: none;` ou `visibility: hidden;`.
- Il ne doit pas être possible de tabuler sur les éventuels éléments interactifs contenus dans les panneaux masqués. Un attribut `tabindex="-1"` doit être rajouté dynamiquement sur chacun de ces éléments. L'attribut doit ensuite être supprimé dès lors que le panneau associé est affiché.
- Les flèches de navigation sont utilisées pour naviguer dans la liste des onglets de pagination et pour sélectionner l'onglet de pagination courant.
- La valeur de l'attribut `aria-selected` doit être modifiée dynamiquement à chaque mise à jour du statut de l'onglet de pagination correspondant.
- La valeur de l'attribut `tabindex` doit également être modifiée dynamiquement chaque fois que le statut de l'onglet de pagination est mis à jour.
- La valeur de l'attribut `aria-hidden` doit être modifiée dynamiquement à chaque mise à jour du statut du panneau correspondant.
- L'alternative textuelle du bouton-image « Arrêter le défilement du carrousel » doit être mise à jour lorsque le bouton-image est activé. Opter alors par exemple pour l'alternative « Relancer le défilement du carrousel ».

Remarque

À noter qu'il est également possible de remplacer les balises `<img />` dans les boutons par du texte, des images vectorielles `<svg>`, ou encore des polices d'icônes.

Composants

Ces composants « Carrousels » sont proposés ici car leur niveau d'accessibilité est jugé bon ou très bon.

Toutefois, avant de les utiliser dans votre projet, il est important de vérifier le respect des spécifications présentées ci-avant. Certains pouvant nécessiter quelques ajustements.

Cases à cocher personnalisées en ARIA

Principe

Les cases à cocher sont des éléments de formulaires qui permettent de sélectionner une ou plusieurs options parmi un groupe d'options proposées.

Les cases à cocher sont dites « personnalisées en ARIA » lorsqu'elles ne sont pas construites sur la base du code HTML standard prévu pour ces éléments par la spécification : `<input type="checkbox" />`.

Le suivi des recommandations ci-dessous permet de reproduire le comportement par défaut des cases à cocher HTML standards, **dans les cas où ces dernières ne peuvent être utilisées**.

Cette fiche s'appuie sur le motif de conception « Checkbox » détaillé dans les WAI-ARIA 1.1 Authoring Practices du W3C.

Socle HTML

```
<p id="question">Question</p>
<ul aria-labelledby="question" role="group">
  <li role="checkbox" aria-checked="false" tabindex="0">
    <img aria-hidden="true" src="checkbox.svg" alt="Non sélectionné : " />
    Choix 1
  </li>
  <li role="checkbox" aria-checked="true" tabindex="0">
    <img aria-hidden="true" src="checkbox-checked.svg" alt="Sélectionné : "
/>
    Choix 2
  </li>
  <li role="checkbox" aria-checked="false" tabindex="0">
    <img aria-hidden="true" src="checkbox.svg" alt="Non sélectionné : " />
    Choix 3
  </li>
</ul>
```

Rôles, états et propriétés ARIA

- `role="group"` doit être appliqué sur le groupe de cases à cocher.
- `role="checkbox"` et `tabindex="0"` doivent être appliqués sur le conteneur de chaque case à cocher.
- `aria-hidden="true"` doit être appliqué sur chaque image simulant une case à cocher.
- L'attribut `aria-checked` doit être appliqué sur le conteneur de chaque case à cocher. Sa valeur doit être renseignée dynamiquement en fonction de l'état de la case à cocher associée :
  - `aria-checked="true"` lorsque la case est cochée.
  - `aria-checked="false"` lorsque la case n'est pas cochée.

- Le groupe de cases à cocher doit être rattaché à l'étiquette du groupe via l'attribut `aria-labelledby` :
    - L'étiquette du groupe doit posséder un attribut `id` renseigné avec une valeur unique.
    - Le groupe de cases à cocher doit posséder un attribut `aria-labelledby` renseigné avec la valeur de l'attribut `id` de l'étiquette du groupe.

Interactions au clavier

Les interactions au clavier sont les mêmes que pour des cases à cocher HTML classiques. À la seule exception que le focus clavier est ici positionné sur le conteneur de la case à cocher, et non uniquement sur la case à cocher.

Espace

Lorsque le focus clavier est sur le conteneur d'une case à cocher, coche/décoche la case à cocher, alternativement et en boucle.

Tab

Permet d'accéder successivement à chaque case à cocher, avant de sortir du groupe, dans l'ordre logique de lecture.

Maj + **Tab**

Même comportement qu'avec la touche **Tab**, mais cette fois dans l'ordre inverse de lecture.

Remarques

La source de l'image, ainsi que son texte alternatif, doivent être modifiés en fonction de l'état de la case à cocher correspondante.

À noter qu'une balise `<img />` est proposée dans l'exemple de code, mais qu'il est également possible de la remplacer par une image vectorielle `<svg>`.

Composants

Ces composants « Cases à cocher personnalisées en ARIA » sont proposés ici car leur niveau d'accessibilité est jugé bon ou très bon.

Toutefois, avant de les utiliser dans votre projet, il est important de vérifier le respect des spécifications présentées ci-avant. Certains pouvant nécessiter quelques ajustements.

Cases à cocher personnalisées en CSS

Principe

Les cases à cocher sont des éléments de formulaires qui permettent de sélectionner une ou plusieurs options parmi un groupe d'options proposées.

Les cases à cocher sont dites « personnalisées en CSS » lorsqu'elles sont construites sur la base du code HTML standard prévu pour ces éléments par la spécification : `<input type="checkbox" />`, mais que ces cases à cocher standards sont masquées à l'écran puis personnalisées en CSS grâce à des images, des polices d'icônes ou des styles spécifiques.

De nombreuses approches sont envisageables. L'exemple ci-dessous expose comment personnaliser des cases à cocher à l'aide de pseudo-éléments CSS et d'images d'arrière-plan, tout en conservant leur caractère accessible.

Socle HTML

```
<fieldset>
  <legend>Question</legend>
  <ul>
    <li>
      <input type="checkbox" id="reponse-1" name="reponse-1" />
      <label for="reponse-1">Réponse 1</label>
    </li>
    <li>
      <input type="checkbox" id="reponse-2" name="reponse-2" />
      <label for="reponse-2">Réponse 2</label>
    </li>
    [...]
  </ul>
</fieldset>
```

Socle CSS

```
label {
  position: relative;
  padding: 0 0 0 2rem;
}

input[type=checkbox] {
  position: absolute;
  opacity: 0;
}

input[type=checkbox] + label::before,
input[type=checkbox] + label::after {
  content: '';
  position: absolute;
  display: inline-block;
}
```

```css
input[type=checkbox] + label::before {
  left: 0.5rem;
  top: 0.15rem;
  width: 0.9rem;
  height: 0.9rem;
  border: 0.05rem solid black;
  background: white;
}

input[type=checkbox]:checked + label::after {
  left: 0.6rem;
  top: 0.28rem;
  height: 0.8rem;
  border-left: 0.8rem solid black;
}

input[type=checkbox]:focus + label::before {
  outline: 0.05rem dotted;
}
```

Infobulles personnalisées

Depuis la rédaction de cette fiche, une mise à jour de la spécification ARIA a eu lieu : passage de ARIA 1.0 (ancienne version) à ARIA 1.1 (nouvelle version).

Cette fiche s'appuie sur les motifs de conception ARIA 1.0 issus du document WAI-ARIA Authoring Practices 1.0, et les recommandations ci-dessous pourront faire l'objet d'ajustements une fois les motifs de conception ARIA 1.1 officialisés.

Pour plus d'informations sur les changements relatifs au passage à ARIA 1.1, consulter les WAI-ARIA Authoring Practices 1.1.

Principe

Les infobulles sont des messages qui permettent d'obtenir une information complémentaire sur un élément. Elles se présentent sous la forme d'un message qui apparaît au survol et à la prise de focus clavier d'un élément.

Les infobulles sont dites « personnalisées » lorsqu'elles ne sont pas construites sur la base du code HTML standard prévu pour ces éléments par la spécification : l'attribut `title`.

Cette fiche s'appuie sur le motif de conception « Tooltip Widget » détaillé dans les WAI-ARIA 1.0 Authoring Practices du W3C.

Socle HTML

```
<a href="#" aria-describedby="infobulle">[Intitulé du lien]</a>
<div role="tooltip" id="infobulle" >[Contenu de l'infobulle]</div>
```

Rôles, états et propriétés ARIA

- `tabindex="0"` doit être appliqué sur l'élément qui permet l'affichage de l'infobulle, si ce dernier n'est pas atteignable au clavier par défaut.
- `role="tooltip"` doit être appliqué sur le conteneur de l'infobulle.
- L'élément qui permet l'affichage de l'infobulle doit être rattaché à l'infobulle via l'attribut `aria-describedby` :
  - Le conteneur de l'infobulle doit posséder un attribut `id` renseigné avec une valeur unique.
  - L'élément qui permet l'affichage de l'infobulle doit posséder un attribut `aria-describedby` renseigné avec la valeur de l'attribut `id` du conteneur de l'infobulle.

Interactions au clavier

Échap

Lorsque l'infobulle est affichée, masque l'infobulle.

Comportements attendus

- L'infobulle doit s'afficher lorsque l'élément qui en permet l'affichage :
  - Est survolé par la souris.
  - Prend le focus clavier.
- L'infobulle doit être masquée lorsque l'élément qui en permet l'affichage :
  - N'est plus survolé par la souris.
  - Perd le focus clavier.
- Une pression de la touche **Échap** doit permettre de masquer l'infobulle.
- L'infobulle doit rester affichée lorsqu'elle est survolée par la souris.
- Lorsque l'infobulle est masquée, elle doit l'être à l'aide de `display: none;` et/ou `visibility: hidden;`. Ou encore supprimée du code source.

Remarque

L'avantage majeur d'une infobulle personnalisée par rapport à son homologue HTML standard (attribut `title`) réside dans le fait que la première est également accessible aux utilisateurs qui naviguent au clavier.

Composants

Ces composants « Infobulles personnalisées » sont proposés ici car leur niveau d'accessibilité est jugé bon ou très bon.

Toutefois, avant de les utiliser dans votre projet, il est important de vérifier le respect des spécifications présentées ci-avant. Certains pouvant nécessiter quelques ajustements.

Menu déroulant

Principe

Un menu déroulant se présente généralement sous la forme d'une liste de boutons accolés qui permettent d'afficher un sous-menu relatif au bouton activé.

Socle HTML

```
<nav role="navigation" aria-label="Menu principal">
  <ul>
    <li>
      <button aria-expanded="false">Bouton de premier niveau dont le sous-menu est masqué</button>
      <ul class="non-visible">
        <li><a href="…">Lien de second niveau</a></li>
        <li><a href="…">Lien de second niveau</a></li>
        […]
      </ul>
    </li>
    <li>
      <button aria-expanded="true">Bouton de premier niveau dont le sous-menu est affiché</button>
      <ul class="visible">
        <li><a href="…">Lien de second niveau</a></li>
        <li><a href="…">Lien de second niveau</a></li>
        […]
      </ul>
    </li>
    <li><a href="…">Lien de premier niveau</a></li>
    […]
  </ul>
</nav>
```

Rôles, états et propriétés ARIA

- La balise `<nav role="navigation">` doit être utilisée pour structurer le menu.
- L'attribut `aria-label` doit être intégré dans cette même balise `<nav role="navigation">` et renseigné avec le nom du menu correspondant.
- Des balises `<ul>` et `<li>` imbriquées doivent être utilisées pour structurer les boutons de premier niveau ainsi que les liens des sous-menus.
- Chaque bouton de premier niveau doit être balisé avec une balise `<button>`.
- L'attribut `aria-expanded` doit être intégré sur chaque bouton de premier niveau. Sa valeur doit être renseignée dynamiquement en fonction de l'état du sous-menu associé :
  - `aria-expanded="false"` lorsque le sous-menu associé est replié.
  - `aria-expanded="true"` lorsque le sous-menu associé est déplié.

Interactions au clavier

Entrée ou **Espace**

- Si le focus clavier est positionné sur un bouton de premier niveau d'un sous-menu replié, déplie le sous-menu associé.
- Si le focus clavier est positionné sur un bouton de premier niveau d'un sous-menu déplié, replie le sous-menu associé.

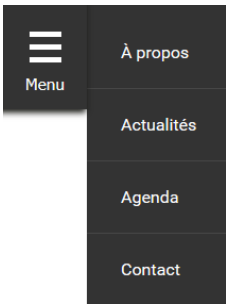
Échap

Si le focus clavier est positionné sur l'un des éléments d'un sous-menu affiché, déplace le focus clavier sur le bouton de premier niveau qui a déclenché l'affichage du sous-menu, puis ferme ce dernier.

Remarque

Les sous-menus non affichés à l'écran doivent être masqués avec `display: none` et/ou `visibility: hidden`.

Menu hamburger



Socle HTML

```
<nav role="navigation" aria-label="Menu principal">
  <button aria-expanded="true">
    <svg aria-hidden="true" focusable="false">[…]</svg>
    Menu
  </button>
  <ul class="visible">
    [Menu de navigation principal]
  </ul>
</nav>
```

Rôles, états et propriétés ARIA

- La balise `<nav role="navigation">` doit être utilisée pour structurer le bouton hamburger **et** le menu.
- L'attribut `aria-label` doit être intégré dans cette même balise `<nav role="navigation">` et renseigné avec le nom du menu correspondant (par exemple `aria-label="Menu principal"`).
- Le bouton hamburger doit être balisé avec une balise `<button>`.
- L'attribut `aria-expanded` doit être appliqué sur le bouton hamburger qui contrôle le menu. Sa valeur doit être renseignée dynamiquement en fonction de l'état du menu :
  - `aria-expanded="true"` lorsque le menu est déplié.
  - `aria-expanded="false"` lorsque le menu est plié.

Interactions au clavier

Entrée et **Espace**

Lorsque le focus clavier est positionné sur le bouton hamburger, affiche/masque alternativement le menu.

Échap

Si le focus clavier est positionné sur l'un des éléments du menu, déplace le focus clavier sur le bouton hamburger qui a déclenché l'affichage du menu, puis ferme ce dernier.

Comportements attendus

- Lorsque le focus clavier est positionné au niveau du bouton hamburger, il est possible de contrôler l'affichage/masquage du menu avec les touches **Espace** et **Entrée**. Pour cela, passer par l'écoute de l'événement `click`.
- Lorsque le menu est plié, il doit être masqué avec `display: none;` et/ou `visibility: hidden;`.
- La valeur de l'attribut `aria-expanded` du bouton hamburger doit être modifiée dynamiquement à chaque changement d'état du menu.

Remarque

Dans le cas particulier où le bouton hamburger n'est pas situé immédiatement avant le code HTML du menu, il est important d'associer techniquement le menu au bouton hamburger qui le contrôle.

Cette association doit être déclarée par l'intermédiaire de l'attribut `aria-controls` :

- Le menu doit posséder un attribut `id` renseigné avec une valeur unique.
- Le bouton hamburger doit posséder un attribut `aria-controls` renseigné avec la valeur de l'attribut `id` du menu.

```
<nav role="navigation" aria-label="Menu principal">
  <button aria-expanded="true" aria-controls="menu-principal">
    <svg aria-hidden="true" focusable="false">[…]</svg>
    Menu
  </button>
  […]
  <ul id="menu-principal" class="visible">
    [Menu de navigation principal]
  </ul>
</nav>
```

Composants

Ces composants « Menu hamburger » sont proposés ici car leur niveau d'accessibilité est jugé bon ou très bon.

Toutefois, avant de les utiliser dans votre projet, il est important de vérifier le respect des spécifications présentées ci-avant. Certains pouvant nécessiter quelques ajustements.

Message d’alerte

Principe

Les messages d’alerte sont un cas particulier de messages de notification qui permettent d’annoncer une erreur ou un avertissement critiques.

Ils attirent l’attention par le biais d’un court message qui apparait spontanément à l’écran, sans rechargement de la page, ni interruption de l’activité.

Cette fiche s’appuie sur le motif de conception « Alert » détaillé dans les WAI-ARIA 1.1 Authoring Practices du W3C.

Socle HTML

Au chargement de la page

```
<div role="alert"></div>
```

Lorsque l’alerte est déclenchée

```
<div role="alert">
  <p>Nous ne pouvons pas vous garantir de place assise sur ce trajet.</p>
</div>
```

Rôles, états et propriétés ARIA

L’attribut `role="alert"` doit être appliqué sur le conteneur du message d’alerte.

Comportements attendus

L’attribut `role="alert"` doit être présent au chargement de la page, de manière statique.

Ce conteneur doit ensuite être peuplé dynamiquement lorsque l’alerte est déclenchée.

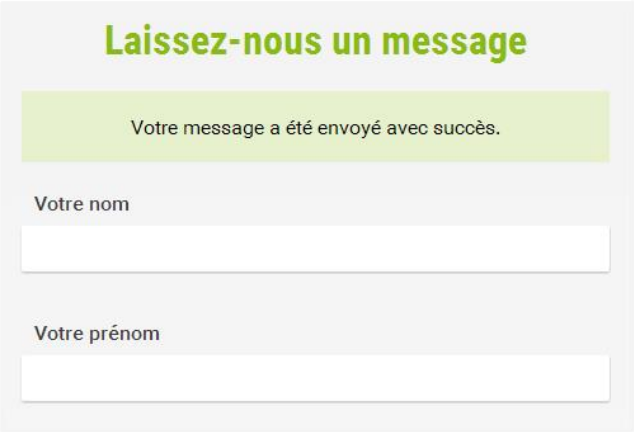
Remarque

Les messages d’alerte ne doivent pas disparaître de l’écran de manière automatique après un certain délai.

Ces messages doivent disparaître seulement sur action manuelle (croix de fermeture, affichage d’une nouvelle page, etc.).

Message de notification

Principe



Les messages de notification sont des composants dynamiques qui permettent d'annoncer une information ou un avertissement non critiques.

Ils attirent l'attention par le biais d'un court message qui apparait spontanément à l'écran, sans rechargement de la page, ni interruption de l'activité.

Socle HTML

Au chargement de la page

```
<div role="status"></div>
```

Lorsque l'alerte est déclenchée

```
<div role="status">
  <p>Votre message a bien été envoyé.</p>
</div>
```

Rôles, états et propriétés ARIA

L'attribut `role="status"` doit être appliqué sur le conteneur du message de notification.

Comportements attendus

L'attribut `role="status"` doit être présent au chargement de la page, de manière statique.

Ce conteneur doit ensuite être peuplé dynamiquement lorsque la notification est déclenchée.

Remarque

Les messages de notification ne doivent pas disparaître de l’écran de manière automatique après un certain délai.

Ces messages doivent disparaître seulement sur action manuelle (croix de fermeture, affichage d’une nouvelle page, etc.).

Onglets

Principe

Les onglets sont des modules dynamiques qui permettent d'optimiser l'affichage du contenu d'une page dans un espace réduit grâce à un système d'étiquettes contrôlant l'affichage de panneaux.

Ils se présentent généralement sous la forme d'une liste d'items accolés qui permettent d'afficher du contenu relatif à l'onglet sélectionné. Un seul onglet peut être activé à la fois.

Cette fiche s'appuie sur le motif de conception « Tabs » détaillé dans les WAI-ARIA 1.1 Authoring Practices du W3C.

Socle HTML

```
<ul role="tablist">
  <li role="tab" id="onglet-1" tabindex="-1" aria-selected="false" aria-
controls="panneau-1">Onglet 1</li>
  <li role="tab" id="onglet-2" tabindex="0" aria-selected="true" aria-
controls="panneau-2">Onglet 2</li>
  <li role="tab" id="onglet-3" tabindex="-1" aria-selected="false" aria-
controls="panneau-3">Onglet 3</li>
  <li role="tab" id="onglet-4" tabindex="-1" aria-selected="false" aria-
controls="panneau-4">Onglet 4</li>
</ul>
<div role="tabpanel" id="panneau-1" aria-labelledby="onglet-1">
tabindex="0">
  [Contenu du premier panneau (masqué)]
</div>
<div role="tabpanel" id="panneau-2" aria-labelledby="onglet-2">
tabindex="0">
  [Contenu du deuxième panneau (affiché, car onglet associé sélectionné)]
</div>
<div role="tabpanel" id="panneau-3" aria-labelledby="onglet-3">
tabindex="0">
  [Contenu du troisième panneau (masqué)]
</div>
<div role="tabpanel" id="panneau-4" aria-labelledby="onglet-4">
tabindex="0">
  [Contenu du quatrième panneau (masqué)]
</div>
```

Rôles, états et propriétés ARIA

- L'attribut `role="tablist"` doit être appliqué sur l'élément qui englobe les onglets.

Dans le cas où les onglets sont orientés verticalement, l'attribut `aria-orientation="vertical"` doit également être appliqué sur cet élément.

- L'attribut `role="tab"` doit être appliqué sur chaque onglet.
- L'attribut `role="tabpanel"` doit être appliqué sur chaque panneau.
- L'attribut `tabindex="0"` doit être appliqué sur chaque panneau.
- Chaque onglet doit être associé à son panneau via l'attribut `aria-controls` :
  - Chaque panneau doit posséder un attribut `id` renseigné avec une valeur unique.
  - Chaque onglet doit posséder un attribut `aria-controls` renseigné avec la valeur de l'attribut `id` de son panneau associé.
- Chaque panneau doit être rattaché à l'onglet qui le contrôle via l'attribut `aria-labelledby` :
  - Chaque onglet doit posséder un attribut `id` renseigné avec une valeur unique.
  - Chaque panneau doit posséder un attribut `aria-labelledby` renseigné avec la valeur de l'attribut `id` de l'onglet qui le contrôle.
- L'attribut `aria-selected` doit être appliqué sur chaque onglet. Sa valeur doit être renseignée dynamiquement en fonction de l'état de l'onglet associé :
  - `aria-selected="true"` sur l'onglet sélectionné.
  - `aria-selected="false"` sur les autres onglets, non sélectionnés.
- L'attribut `tabindex` doit être appliqué sur chaque onglet. Sa valeur doit être renseignée dynamiquement en fonction de l'état de l'onglet associé :
  - `tabindex="0"` sur l'onglet sélectionné.
  - `tabindex="-1"` sur les autres onglets, non sélectionnés.

Interactions au clavier

Tab et **Maj** + **Tab**

Lorsque l'on accède en tabulant aux onglets, le focus est positionné au niveau de l'onglet sélectionné dans le groupe d'onglets. Lorsque le focus est positionné sur un onglet, la prochaine tabulation permet de quitter le groupe d'onglets et de se positionner sur le panneau affiché.

Flèche gauche

Lorsque le focus est positionné sur un onglet, déplace le focus clavier vers le précédent onglet du groupe et sélectionne cet onglet. Si le focus clavier est positionné au niveau du premier onglet du groupe au moment où la touche est pressée, le focus clavier est déplacé au niveau du dernier onglet du groupe et cet onglet est sélectionné.

Dans le cas où les onglets sont orientés verticalement, prévoir également ce comportement pour la touche **Flèche haut**.

Flèche droite

Lorsque le focus est positionné sur un onglet, déplace le focus clavier vers l'onglet suivant dans le groupe et sélectionne cet onglet. Si le focus clavier est positionné au niveau du dernier onglet du groupe au moment où la touche est pressée, le focus clavier est déplacé au niveau du premier onglet du groupe et cet onglet est sélectionné.

Dans le cas où les onglets sont orientés verticalement, prévoir également ce comportement pour la touche **Flèche bas**.

Remarque

Les panneaux non affichés à l'écran doivent être masqués avec `display: none` et/ou `visibility: hidden`.

Composants

Ces composants « Onglets » sont proposés ici car leur niveau d'accessibilité est jugé bon ou très bon.

Toutefois, avant de les utiliser dans votre projet, il est important de vérifier le respect des spécifications présentées ci-avant. Certains pouvant nécessiter quelques ajustements.

Panneaux dépliants

Principe

Les panneaux dépliants sont des composants dynamiques qui permettent d'optimiser l'affichage d'un contenu dans un espace réduit grâce à un système de « plier / déplier ».

Ils sont contrôlables le plus souvent via une action sur un bouton qui surplombe le contenu du panneau.

Cette fiche s'appuie sur le motif de conception « Disclosure » détaillé dans les WAI-ARIA 1.1 Authoring Practices du W3C.

Socle HTML

```
<button aria-expanded="true">[Intitulé du bouton]<button>
<div class="visible">
  <p>[Contenu du panneau dépliant]</p>
</div>
```

Rôles, états et propriétés ARIA

L'attribut `aria-expanded` doit être appliqué sur le bouton qui contrôle le panneau dépliant. Sa valeur doit être renseignée dynamiquement en fonction de l'état du panneau dépliant associé :

- `aria-expanded="true"` lorsque le panneau est déplié.
- `aria-expanded="false"` lorsque le panneau est plié.

Interactions au clavier

Entrée et **Espace**

Lorsque le focus clavier est positionné sur le bouton, affiche/masque alternativement le panneau dépliant associé.

Comportements attendus

- Lorsque le focus clavier est positionné au niveau du bouton, il est possible de contrôler l'affichage/masquage du panneau associé avec les touches **Espace** et **Entrée**. Pour cela, passer par l'écoute de l'événement `click`.
- Lorsque le panneau est plié, il doit être masqué avec `display: none;` et/ou `visibility: hidden;`.

- La valeur de l'attribut `aria-expanded` doit être modifiée dynamiquement à chaque changement d'état du panneau dépliant.

Remarque

Dans le cas particulier où le bouton d'action n'est pas situé immédiatement avant le code HTML du panneau dépliant associé, il est important d'associer techniquement le panneau au bouton qui le contrôle.

Cette association doit être déclarée par l'intermédiaire de l'attribut `aria-controls` :

- Le panneau dépliant doit posséder un attribut `id` renseigné avec une valeur unique.
- Le bouton doit posséder un attribut `aria-controls` renseigné avec la valeur de l'attribut `id` du panneau dépliant associé.

```
<button aria-expanded="true" aria-controls="panneau-depliant">[Intitulé du bouton]<button>
[...]
<div id="panneau-depliant" class="visible">
  <p>[Contenu du panneau dépliant]</p>
</div>
```

Composants

Ces composants « Panneaux dépliants » sont proposés ici car leur niveau d'accessibilité est jugé bon ou très bon.

Toutefois, avant de les utiliser dans votre projet, il est important de vérifier le respect des spécifications présentées ci-avant. Certains pouvant nécessiter quelques ajustements.

Sliders personnalisés

Principe

Les sliders sont des éléments de formulaires qui permettent de sélectionner une valeur unique en déplaçant un curseur sur une échelle graduée.

Les sliders sont dits « personnalisés » lorsqu'ils ne sont pas construits sur la base du code HTML standard prévu pour ces éléments par la spécification : `<input type="range" />`.

Le suivi des recommandations ci-dessous permet de reproduire le comportement par défaut des sliders HTML standards, **dans les cas où ces derniers ne peuvent être utilisés**.

Cette fiche s'appuie sur le motif de conception « Slider » détaillé dans les WAI-ARIA 1.1 Authoring Practices du W3C.

Socle HTML

```
<p id="etiquette">Étiquette du slider</p>
<button role="slider" aria-valuemin="0" aria-valuemax="10" aria-
valuenow="8" aria-labelledby="etiquette">
  <img src="curseur.png" aria-hidden="true" alt="" />
</button>
[…]
```

Rôles, états et propriétés ARIA

- `role="slider"` doit être appliqué sur le curseur.
- L'attribut `aria-valuemin` doit être appliqué sur le curseur. Cet attribut doit être renseigné avec la valeur minimale autorisée pour le slider.
- L'attribut `aria-valuemax` doit être appliqué sur le curseur. Cet attribut doit être renseigné avec la valeur maximale autorisée pour le slider.
- L'attribut `aria-valuenow` doit être appliqué sur le curseur. Sa valeur doit être renseignée dynamiquement en reprenant celle de la position courante du curseur.
- Le curseur doit être rattaché à son étiquette via l'attribut `aria-labelledby` :
  - L'étiquette du slider doit posséder un attribut `id` renseigné avec une valeur unique.
  - Le curseur doit posséder un attribut `aria-labelledby` renseigné avec la valeur de l'attribut `id` de l'étiquette du slider.
- `aria-hidden="true"` doit être appliqué sur l'image simulant le curseur.

Interactions au clavier

Les interactions au clavier sont les mêmes que pour les sliders HTML classiques. À la seule exception que le focus clavier est ici positionné directement sur le curseur et non sur l'ensemble du slider.

Tab

Lorsque l'on accède en tabulant au slider, le focus est positionné au niveau du curseur. Lorsque le focus est positionné sur le curseur, la prochaine tabulation permet de quitter le slider.

Maj + **Tab**

Même comportement qu'avec la touche **Tab**, mais cette fois dans l'ordre inverse de lecture.

Flèche haut et **Flèche droite**

Lorsque le focus est positionné sur le curseur, déplace le curseur et incrémente la valeur du slider d'un pas.

Flèche bas et **Flèche gauche**

Lorsque le focus est positionné sur le curseur, déplace le curseur et décrémente la valeur du slider d'un pas.

Origine

Lorsque le focus est positionné sur le curseur, déplace le curseur et décrémente la valeur du slider à son minimum.

Fin

Lorsque le focus est positionné sur le curseur, déplace le curseur et incrémente la valeur du slider à son maximum.

Page précédente

Lorsque le focus est positionné sur le curseur, déplace le curseur et incrémente la valeur du slider d'un nombre prédéfini de pas.

Page suivante

Lorsque le focus est positionné sur le curseur, déplace le curseur et décrémente la valeur du slider d'un nombre prédéfini de pas.

Comportements attendus

- Le focus clavier est positionné sur le curseur lors de l'utilisation du slider et y reste jusqu'à l'utilisation de la touche **Tab**.
- Lorsque le focus clavier est positionné sur le curseur, il est possible de modifier la valeur du slider en utilisant les touches **Flèche haut**, **Flèche bas**, **Origine**, **Fin**, **Page précédente** et **Page suivante**.
- La valeur de l'attribut `aria-valuenow` doit être modifiée dynamiquement à chaque mise à jour du slider et doit être identique à la valeur du slider.
- La valeur du slider ne peut être supérieure à la valeur renseignée dans l'attribut `aria-valuemax`.
- La valeur du slider ne peut être inférieure à la valeur renseignée dans l'attribut `aria-valuemin`.

Remarque

Le choix du nombre de pas à « sauter » lors de la pression sur les touches **Page précédente** et **Page suivante** est libre.

D'autre part, il est important de noter que l'information numérique véhiculée par l'attribut `aria-valuenow` n'est pas toujours suffisamment explicite. C'est par exemple le cas lorsqu'un slider permet de sélectionner l'un des sept jours de la semaine :

```
<p id="evenement-jour">Jour de la semaine</p>
<button role="slider" aria-valumin="0" aria-valuemax="6" aria-valuenow="3"
aria-labelledby="evenement-jour">
  <img src="curseur.png" aria-hidden="true" alt="" />
</button>
[…]
```

Dans ces situations, l'attribut optionnel `aria-valuetext` doit être utilisé en parallèle, afin de traduire la valeur courante du slider de manière plus compréhensible :

```
<p id="evenement-jour">Jour de la semaine</p>
<button role="slider" aria-valumin="0" aria-valuemax="6" aria-valuenow="3"
aria-valuetext="Jeudi" aria-labelledby="evenement-jour">
  <img src="curseur.png" aria-hidden="true" alt="" />
</button>
[…]
```

S'il est utilisé, la valeur de l'attribut `aria-valuetext` doit également être renseignée dynamiquement à chaque mise à jour de la position du curseur.

À noter enfin qu'une balise `<img />` est proposée dans l'exemple de code, mais qu'il est également possible de la remplacer par une image vectorielle `<svg>` ou une police d'icônes.

Composants

Ces composants « Sliders personnalisés » sont proposés ici car leur niveau d’accessibilité est jugé bon ou très bon.

Toutefois, avant de les utiliser dans votre projet, il est important de vérifier le respect des spécifications présentées ci-avant. Certains pouvant nécessiter quelques ajustements.

Spinbuttons personnalisés

Principe

Les spinbuttons sont des éléments de formulaires qui permettent de renseigner une valeur numérique. Ils se présentent sous la forme d'un champ de texte associé à deux boutons permettant respectivement d'augmenter et de diminuer d'un pas la valeur du champ.

Les spinbuttons sont dits « personnalisés » lorsqu'ils ne sont pas construits sur la base du code HTML standard prévu pour ces éléments par la spécification : `<input type="number" />`, mais par l'intermédiaire d'un champ de texte associé à deux images, polices d'icônes ou styles spécifiques, pour afficher les « boutons » d'incrémentation et de décrémentation.

Le suivi des recommandations ci-dessous permet de reproduire le comportement par défaut des spinbuttons HTML standards, **dans les cas où ces derniers ne peuvent être utilisés**.

Cette fiche s'appuie sur le motif de conception « Spinbutton » détaillé dans les WAI-ARIA 1.1 Authoring Practices du W3C.

Socle HTML

```
<p id="etiquette">Étiquette du spinbutton</p>
<div>
  <input type="text" role="spinbutton" aria-valumin="0" aria-valuemax="10"
aria-valuenow="8" aria-labelledby="etiquette" />
  <img aria-hidden="true" src="plus.png" alt="+" />
  <img aria-hidden="true" src="minus.png" alt="-" />
</div>
```

Rôles, états et propriétés ARIA

- `role="spinbutton"` doit être appliqué sur le spinbutton.
- L'attribut `aria-valuemin` doit être appliqué sur le spinbutton. Cet attribut doit être renseigné avec la valeur minimale autorisée pour le spinbutton.
- L'attribut `aria-valuemax` doit être appliqué sur le spinbutton. Cet attribut doit être renseigné avec la valeur maximale autorisée pour le spinbutton.
- L'attribut `aria-valuenow` doit être appliqué sur le spinbutton. Sa valeur doit être renseignée dynamiquement en reprenant celle du spinbutton.
- Le spinbutton doit être rattaché à son étiquette via l'attribut `aria-labelledby` :
  - L'étiquette du spinbutton doit posséder un attribut `id` renseigné avec une valeur unique.
  - Le spinbutton doit posséder un attribut `aria-labelledby` renseigné avec la valeur de l'attribut `id` de l'étiquette du spinbutton.

- `aria-hidden="true"` doit être appliqué sur chaque image simulant un bouton d'incrémentation ou de décrémentation.

Interactions au clavier

Les interactions au clavier sont les mêmes que pour les spinbuttons HTML classiques.

Tab

Lorsque l'on accède en tabulant au spinbutton, le focus est positionné au niveau du champ de texte. Lorsque le focus est positionné sur le champ de texte, la prochaine tabulation permet de quitter le spinbutton.

Maj + **Tab**

Même comportement qu'avec la touche **Tab**, mais cette fois dans l'ordre inverse de lecture.

Flèche haut

Lorsque le focus est positionné dans le champ de texte, incrémente la valeur du champ de texte d'un pas.

Flèche bas

Lorsque le focus est positionné dans le champ de texte, décrémente la valeur du champ de texte d'un pas.

Origine

Lorsque le focus est positionné dans le champ de texte, décrémente la valeur du spinbutton à son minimum.

Fin

Lorsque le focus est positionné dans le champ de texte, incrémente la valeur du spinbutton à son maximum.

Comportements attendus

- Le focus clavier est positionné dans le champ de texte lors de l'utilisation du spinbutton et y reste jusqu'à l'utilisation de la touche **Tab**.
- Lorsque le focus clavier est positionné dans le champ de texte, il est possible de modifier la valeur du spinbutton à la fois en la renseignant directement au clavier ou en utilisant les touches **Flèche haut**, **Flèche bas**, **Origine** et **Fin**.

- Les boutons d’incrémentation et de décrémentation ne sont pas atteignables au clavier, mais sont fonctionnels à la souris.
- La valeur de l’attribut `aria-valuenow` doit être modifiée dynamiquement à chaque mise à jour du spinbutton et doit être identique à la valeur du spinbutton.
- La valeur du spinbutton ne peut être supérieure à la valeur renseignée dans l’attribut `aria-valuemax`.
- La valeur du spinbutton ne peut être inférieure à la valeur renseignée dans l’attribut `aria-valuemin`.