Résumé des points d'attention en accessibilité

Auteur : ÉTIENNE COLLET

Relecteur : ? ?

Résumé

Cet article résume comment mettre en place un site web accessible, quels sont les points d'attention ainsi que les tests à réaliser. Il se base sur les supports [1] d'une formation de trois jours délivrée par Témésis en 12/2024.

Table des matières

1 Attributs ARIA courants **1**

2 Structure globale d'une page **1**

3 Présentation des contenus **1**

3.1 Généralités . . . 1

3.1.1 Navigation clavier . . . 2

3.2 Structures de contenu simples . . . 2

3.2.1 Sémentique . . . 2

3.2.2 Lien . . . 2

3.2.3 Images et SVG . . . 2

3.3 Structures de contenu spécifiques . . . 2

3.3.1 Formulaires . . . 2

3.3.2 Tableaux . . . 3

3.3.3 Iframe . . . 3

3.3.4 Vidéo et audio . . . 3

4 Composants **3**

4.1 Disclosure . . . 3

4.2 Et plein d'autre 3

5 Les Outils **3**

5.1 Les extentions de navigateur . . . 3

5.2 Intégration continue . . . 3

GLOBALIS
ACCÉLÉRATEUR DE PROJETS DIGITAUX

1 Attributs ARIA courants

L'API ARIA permet aux sites web « dynamisés » en JS d'être plus accessibles aux personnes utilisant des Technologie d'Assistance (TA). C'est une surcouche sémantique passive qui peut ajouter des noms, rôles, états et propriétés à des composants d'interfaces. Cela comble certaines limites du HTML en exposant ses propriétés à l'arbre d'accessibilité. En voici les éléments les plus importants :

- `aria-label` : redéfinit le nom d'un élément via une chaîne de caractères.
- `aria-labelledby` : redéfinit le nom d'un élément via des identifiants (id).
- `aria-describedby` : fournit des descriptions supplémentaires via des identifiants.
- `aria-atomic` : indique si l'ensemble du contenu d'un élément doit être considéré comme une seule entité par les TA lorsque le contenu change dynamiquement.
- `aria-current` : indique que cet élément représente l'élément actuel au sein d'un ensemble apparenté.
- `aria-live` : indique qu'un élément doit être annoncé aux TA lorsque son contenu change.
- `aria-level` : indique le niveau hiérarchique d'un élément (accompagné d'un `role="heading"`).
- `aria-disabled` : indique si un élément est désactivé et non interactif.
- `aria-expanded` : indique si un élément repliable est ouvert ou fermé.
- `aria-modal` : indique aux TA que les fenêtres situées sous la fenêtre modale ne sont pas disponibles pour des interactions.
- `aria-pressed` : indique l'état actuel d'un bouton à bascule (activé/désactivé).
- `aria-required` : indique qu'un champ de saisie est obligatoire et doit être rempli avant de soumettre un formulaire.
- `aria-selected` : indique si une option au sein d'un élément sélectionnable est actuellement sélectionnée.
- `aria-hidden` : indique si un élément doit être visible ou caché pour les technologies d'assistance.
- `aria-invalid` : indique si la valeur d'un champ de saisie (formulaire) est valide ou non.

(Liste plus exhaustive)

Disclaimer : Toujours privilégier la sémantique HTML par rapport à ARIA.

2 Structure globale d'une page

Pour faciliter la navigation dans le site, au moins deux systèmes de navigation sont requis parmi les suivants :

- `Menu de navigation`
- `Plan du site`
- `Moteur de recherche`

Les menus de navigation et les barres de navigation doivent être présents à la même place dans la présentation, pour un ensemble de pages données.

Exemple de structure HTML :

```html
<html lang="fr" dir="ltr">
  <head>
    <meta
      name="viewport"
      content="width=device-width, initial-scale=1.0"
    />
    <title>Page | Site</title>
  </head>
  <body>
    <a class="sr-only sr-only-focusable" href="#main">
      Allez au contenu
    </a>
    <header role="banner">
      <!-- Contenu de l'entête -->
      <nav role="navigation" aria-label="menu principal">
        <!-- Menu de navigation -->
      </nav>
    </header>
    <main id="main" role="main">
      <!-- Contenu principal -->
    </main>
    <aside role="complementary">
      <!-- Contenu complémentaire -->
    </aside>
    <footer role="contentinfo">
      <!-- Pied de page -->
    </footer>
  </body>
</html>
```

3 Présentation des contenus

3.1 Généralités

Le contraste minimum recommandé est de `4,5` (il peut être inférieur dans certains cas précis). Pour le texte se trouvant par-dessus une image ajoutée en CSS, ne pas oublier de mettre une couleur par défaut, exemple :

```css
.back {
  background: #153d7c url(texture-blue.png) repeat 0 0;
}
```

Par ailleurs, pour cacher visuellement un élément tout en le rendant accessible aux TA, il est conseillé d'utiliser cette classe :

```css
.sr-only {
  border: 0 !important;
  clip: rect(1px, 1px, 1px, 1px) !important;
  -webkit-clip-path: inset(50%) !important;
  clip-path: inset(50%) !important;
  height: 1px !important;
  margin: -1px !important;
  overflow: hidden !important;
  padding: 0 !important;
  position: absolute !important;
  width: 1px !important;
  white-space: nowrap !important;
}
```

De plus, pour avoir un texte caché visuellement mais visible lors de la prise de focus, il est recommandé d'utiliser cette classe :

```
.sr-only-focusable:focus,
.sr-only-focusable:active {
  clip: auto !important;
  -webkit-clip-path: none !important;
  clip-path: none !important;
  height: auto !important;
  margin: auto !important;
  overflow: visible !important;
  width: auto !important;
  white-space: normal !important;
}
```

Disclaimer : Aucune information porteuse de sens ne doit être transmise uniquement avec du style. Elles doivent être complétées d'un `aria-current="true"` ou d'un texte caché mais restitué qui indique l'élément courant (ex : `<span class="sr-only"> - active</span>`).

3.1.1 Navigation clavier

- Principes généraux
  - Les composants interactifs doivent être utilisables au clavier.
  - Toute action réalisable avec un curseur ou au toucher doit être possible au clavier.
  - Privilégier les éléments nativement focusables (liens, boutons, champs de formulaire).
  - Toujours déplacer le focus là où cela fait sens (par exemple, après une interaction ou pour poursuivre un chemin logique).
- Gestion de l'attribut `tabindex`
  - Utiliser `tabindex="0"` pour rendre un élément non focusable nativement tabulable dans l'ordre du DOM.
  - Utiliser `tabindex="-1"` pour :
    - → Rendre un élément focusable uniquement via JavaScript (par exemple, `<div>`, `<h1>`).
    - → Empêcher la prise de focus sur un élément nativement tabulable.

Disclaimer : Éviter les valeurs positives (`tabindex="1"` et plus) car elles perturbent l'ordre logique de tabulation et créent de la confusion pour les utilisateurs.

- Visibilité du focus
  - Le focus sur les éléments interactifs doit être toujours visible.
  - En dernier recours, utilisez la règle CSS suivante pour éviter la suppression intempestive des contours de focus :

```
*:focus:not(:focus-visible) {
    outline: none !important;
}
```

Disclaimer : Lors de la tabulation, l'ordre des éléments qui prennent le focus est logique et l'enchaînement de la lecture reste cohérent dans les deux sens pour une personne qui ne verrait pas l'écran, par exemple.

3.2 Structures de contenu simples

3.2.1 Sémentique

L'utilisation de balises HTML sémantiques est essentielle pour garantir une structure accessible. Les balises `<h1>` à `<h6>` doivent être utilisées pour les titres (sans sauter de niveaux si possible), `<p>` pour les paragraphes, et `<ul>` ou `<ol>` pour les listes. Il est déconseillé d'utiliser des balises génériques comme `<div>` ou `<span>` pour remplacer des éléments sémantiques structurés. Pour des cas particuliers, il est possible d'ajouter des rôles ARIA, tels que `role="heading"` et `aria-level="n"` afin de renforcer l'accessibilité et la compréhension par les technologies d'assistance.

3.2.2 Lien

Les liens doivent toujours inclure un attribut `href` avec une valeur non vide pour être interprétés correctement. Il est crucial de fournir des intitulés explicites permettant aux utilisateurs de comprendre leur destination ou leur fonction. Pour ce faire, on peut utiliser l'attribut `aria-label`, ou bien cacher un texte supplémentaire en utilisant la classe `sr-only`. Lorsque les liens font partie d'un ensemble interactif, il est recommandé d'étendre la zone cliquable avec des techniques CSS surtout pour l'affichage responsive.

3.2.3 Images et SVG

- Ajouter des attributs `alt` descriptifs et détaillé pour toutes les images informatives.
- Laisser un `alt=""` vide pour les images décoratives.
- Pour les SVG décoratifs, utiliser `aria-hidden="true"`.
- Associer les images informatives avec du texte alternatif visible ou caché à l'aide de la classe `sr-only`.
- Utiliser des légendes et des balises `<figure>` et `<figcaption>` pour les galeries.

3.3 Structures de contenu spécifiques

3.3.1 Formulaires

Disclaimer : Beaucoup de points d'attention

- Associer les étiquettes (`<label>`) aux champs via l'attribut `for` et l'ID correspondant.
- Marquer les champs obligatoires avec `required` ou `aria-required="true"`.
- L'attribut `placeholder` ne doit pas être utilisé comme alternative à un label.
- Utiliser `autocomplete` pour faciliter la saisie des informations. Les valeurs possibles.
- Grouper les champs liés avec `<fieldset>` et `<legend>` ou utiliser `role="group"` avec des labels ARIA. La seconde solution permet d'imbriquer des groupes de champs.

- Ajouter des indications sur les formats attendus (aide à la saisie) via des descriptions directement dans le label ou avec `aria-describedby`.
- Pour la gestion des erreurs, il y a deux possibilités :

  → Un bloc en haut du formulaire avec toutes les erreurs. Pour chaque erreur, une ancre cliquable déplace le focus sur le champ correspondant.

  → Chaque champ en erreur contient une description claire des erreurs qui lui sont associées avec, dans certains cas, des exemples de données valides.

  Disclaimer : Lors de l'apparition d'une erreur à la soumission, le focus se déplace sur le premier bloc d'erreur et le titre de la page change pour indiquer l'erreur. De plus, on met `aria-invalid="true"` sur les champs en erreur.

3.3.2 Tableaux

Un tableau doit avoir un titre avec `<caption>` et des entêtes définis par `<th>`. Les tableaux complexes nécessitent des attributs `scope` ou `headers` pour relier les cellules aux entêtes. Pour un tableau décoratif, utiliser `role="presentation"` pour qu'il soit ignoré par les technologies d'assistance.

3.3.3 Iframe

Les iframes doivent inclure un attribut **title** descriptif pour expliquer leur contenu. Les iframes techniques, comme celles pour le suivi, doivent être marquées avec `aria-hidden="true"` ou **hidden** pour être ignorées par les lecteurs d'écran.

3.3.4 Vidéo et audio

Une vidéo ou un audio dois fournir des contrôles accessibles pour la lecture, la pause et le volume. La présence de sous-titres et d'audiodescriptions est obligatoire lorsque des informations critiques sont transmises visuellement ou auditivement. il faut inclure des transcriptions textuelles si l'audiodescription est absente. Enfin, il existe déjà des lecteurs conformes aux normes d'accessibilité (ex : AblePlayer, Plyr).

4 Composants

Il est possible de retrouver une liste d'exemple de bonne pratique pour les composant dynamique ici : https://www.w3.org/WAI/ARIA/apg/patterns/

Disclaimer : Il y a d'autre moyen de faire pour chaque composant tous en restant accessible. Il ne faut pas se limité a cela !

4.1 Disclosure

Il faut bien changer `aria-expanded=\true"` quand le contenue est afficher

```
<button type="button" aria-expanded="false/true">
  5 consiels de tech-lead
</button>
<div id="panel" class="hidden/visible">
  <p>...</p>
</div>
```

Disclaimer : Si le contenu apparaît avant le bouton :
- ajouter dynamiquement l'attribut `tabindex="-1"` sur le contenu nouvellement affiché.
- positionner le focus dessus.

Disclaimer : Pas de scroll infini

4.2 Et plein d'autre ...

5 Les Outils

5.1 Les extentions de navigateur

— **[F12]** - DevTools (Chrome) / Developer Tools (Firefox)
— **HeadingsMap** (extension navigateur) - Firefox / Chrome
— **Total11y** (extension navigateur) - khan.github.io/tota11y/
— **Web Developer** (extension navigateur) - chrispederick.com/work/web-developer/
— **Wave** (extension navigateur) - wave.webaim.org/extension/
— **Accessibility Insight** (extension Chrome) - accessibilityinsights.io/
— **ARC Toolkit** (extension Chrome) - paciellogroup.com/toolkit/
— **Color Contrast Analyser** (application bureau) - developer.paciellogroup.com/resources/contrastanalyser/
— **WCAG contrast checker** (extension navigateur) - Firefox / Chrome
— **Stylus** (extension navigateur) - Firefox / Chrome

5.2 Intégration continue

— **A11ygato (Orange)** - Orange-OpenSource/a11ygato-platform
— **Pa11y** - pa11y.org
— **Tanaguru** - github.com/Tanaguru/Tanaguru
— **WebHint** - webhint.io/docs/user-guide/
— **Tenon** - tenon.io/getcode.php
— **Comparatif (2018)** - alphagov.github.io/accessibility-tool-audit/
— **Projet aXe-core (Deque)** - axe-core

Références

[1] Témésis. Guide d'accessibilité. `https://drive.google.com/file/d/1zeaJyDAP7umaMLj1yfGaxGwr_ypxMRHr/view?usp=sharing`, 2024. Consulté en décembre 2024.