

Guide détaillé des bonnes pratiques pour améliorer l'accessibilité des applications (Vue.js / Vuetify).

I. Introduction à l'accessibilité web

Qu'est-ce que l'accessibilité numérique ?

L'accessibilité web consiste à **rendre les sites et les applications web utilisables par tous les individus**, quelles que soient leurs capacités ou leurs limitations. Elle vise à éliminer les barrières qui peuvent empêcher certaines personnes de naviguer, d'interagir et de bénéficier pleinement des contenus en ligne. L'accessibilité web ne concerne pas uniquement les personnes atteintes de handicaps, mais aussi les personnes âgées, les personnes avec une connexion internet lente, ou celles qui utilisent des dispositifs d'assistance tels que des lecteurs d'écran. Cela implique de prendre en compte les différentes capacités et limitations des utilisateurs, notamment en termes de **vision, d'audition, de motricité et de cognition**.

Pourquoi l'accessibilité numérique est-elle importante ?

1. Inclusion et respect des droits fondamentaux : L'accessibilité numérique est une question de respect des droits fondamentaux des personnes handicapées. En rendant les sites web accessibles, vous permettez à **tous les individus** de participer pleinement à la société en ligne, d'accéder à l'information, d'interagir avec les services en ligne et de bénéficier des opportunités offertes par le monde numérique.

2. Avantages commerciaux : L'accessibilité numérique peut avoir des avantages commerciaux tangibles. En rendant votre site web accessible, vous **élargissez votre public cible**, ce qui peut augmenter votre visibilité, votre trafic et votre clientèle potentielle. En rendant l'expérience utilisateur plus agréable pour tous, vous **améliorez également la réputation de votre entreprise** et **favorisez la fidélité des clients**.

3. SEO (Search Engine Optimization) : L'accessibilité numérique est étroitement liée à l'optimisation des moteurs de recherche. Les moteurs de recherche, tels que Google, accordent une **importance croissante à l'accessibilité et à l'expérience utilisateur**.

pour classer les sites web dans leurs résultats de recherche. Un site web accessible et convivial pour tous les utilisateurs a de meilleures chances d'obtenir un meilleur classement dans les résultats de recherche, ce qui peut augmenter la visibilité et le trafic organique vers votre site.

4. Obligations légales : Dans de nombreux pays, il existe des **lois et des réglementations qui exigent que les sites web soient accessibles**. Par exemple, aux États-Unis, la Section 508 du Rehabilitation Act exige que les sites web fédéraux soient accessibles aux personnes handicapées. En Europe, la Directive européenne sur l'accessibilité des sites web et des applications mobiles des organismes du secteur public (2016/2102/UE) impose également des exigences d'accessibilité pour les sites web publics. Ne pas respecter ces obligations légales peut entraîner des conséquences juridiques, des amendes et des réputations négatives pour les entreprises.

II. Principes de l'accessibilité web

Les normes et lignes directrices d'accessibilité (WCAG)

Les normes les plus largement reconnues et utilisées en matière d'accessibilité web sont les **WCAG** (Web Content Accessibility Guidelines) éditées par le **W3C** (World Wide Web Consortium). Les WCAG fournissent des recommandations détaillées pour rendre les contenus web accessibles à un large éventail d'utilisateurs.

Les WCAG sont organisées en quatre principes fondamentaux :

1. **Perceptible** : Les informations et les composants de l'interface utilisateur doivent être perceptibles par tous les utilisateurs, quel que soit leur mode de perception (visuel, auditif, tactile, etc.).
2. **Utilisable** : Les utilisateurs doivent pouvoir interagir avec les contenus et les composants de manière intuitive et sans confusion.
3. **Compréhensible** : Les contenus et l'interface utilisateur doivent être clairs et compréhensibles pour tous les utilisateurs, en évitant les jargons complexes et les éléments confus.
4. **Robuste** : Les contenus doivent être conçus de manière à fonctionner de manière fiable sur différentes technologies et dispositifs.

Ces principes sont ensuite déclinés en critères de succès, **classés par niveaux de conformité** : **A** (le niveau minimum), **AA** (recommandé) et **AAA** (le niveau le plus élevé). Les WCAG fournissent des directives spécifiques pour chaque critère de succès, couvrant des aspects tels que la structure du contenu, la navigation, les contrôles interactifs, le contraste des couleurs, etc.

Les principes de base de l'accessibilité web

1. **Utilisez des balises sémantiques** : Assurez-vous d'utiliser les balises HTML appropriées pour structurer votre contenu. Utilisez les balises <header>, <nav>, <main>, <section>, <article>, <aside>, <footer>, etc., de manière appropriée pour décrire la structure de la page.

En Vue.js avec Vuetify, nous avons donc des balises sémantiques tels que :

<v-app> : La balise racine de votre application Vuetify. Elle englobe tous les autres composants.

<v-header> : Utilisé pour la section d'en-tête de votre application.

<v-nav> : Utilisé pour la barre de navigation de votre application.

<v-main> : Utilisé pour le contenu principal de votre application.

<v-article> : Utilisé pour des sections de contenu indépendantes et autonomes, telles que des articles de blog.

<v-aside> : Utilisé pour le contenu supplémentaire, comme une barre latérale ou des informations complémentaires.

<v-footer> : Utilisé pour le pied de page de votre application.

<v-section> : Utilisé pour définir une section distincte de contenu.

<v-container> : Utilisé pour créer un conteneur de mise en page centré et réactif pour votre contenu.

<v-row> : Utilisé pour créer une ligne de grille dans laquelle vous pouvez placer des colonnes.

<v-col> : Utilisé pour créer des colonnes dans une grille pour organiser votre contenu de manière flexible.

<v-form> : Utilisé pour créer un formulaire.

<v-card> : Utilisé pour créer des cartes pour afficher du contenu structuré.

<v-list> : Utilisé pour créer des listes d'éléments.

<v-dialog> : Utilisé pour créer des boîtes de dialogue ou des modales.


<v-menu> : Utilisé pour créer des menus déroulants ou des menus contextuels.

<v-tooltip> : Utilisé pour ajouter des info-bulles pour fournir des informations supplémentaires sur un élément.


<v-progress-linear> : Utilisé pour afficher une barre de progression linéaire.

<v-slider> : Utilisé pour créer des curseurs interactifs pour les valeurs numériques.

2. **Ajoutez des descriptions alternatives pour les images** : Utilisez l'attribut **alt** pour fournir une **description alternative aux images**. Cela aide les utilisateurs malvoyants à comprendre le contenu de l'image à l'aide de lecteurs d'écran. Vous pouvez également envisager d'utiliser l'attribut **aria-label** pour donner plus d'informations.
3. **Utilisez des couleurs contrastées et des indicateurs visuels** : Assurez-vous que le **texte** et les **éléments interactifs** ont un **contraste suffisant par rapport à l'arrière-plan**. Cela aide les personnes ayant une vision réduite ou des problèmes de dyslexie à lire le contenu plus facilement. Vous pouvez utiliser des outils en ligne pour vérifier le contraste des couleurs, en respectant les recommandations du WCAG (Web Content Accessibility Guidelines), tel que "Contrast Checker" de WebAIM.
En plus du contraste des couleurs, utilisez des **motifs**, des **formes** ou d'autres indicateurs visuels pour transmettre des **informations importantes**. Par exemple, utilisez des **icônes** ou des **flèches directionnelles** pour indiquer les **états** ou les **actions**, afin que les utilisateurs aveugles ou malvoyants puissent également comprendre ces éléments.
4. **Accessibilité des formulaires** : Assurez-vous que vos formulaires sont accessibles en utilisant des balises HTML appropriées pour les champs de saisie, les étiquettes, les groupes de boutons, etc. Utilisez les attributs ARIA nécessaires pour fournir des informations supplémentaires lorsque cela est nécessaire.
5. **Utilisez un focus visuel clair** : Assurez-vous que les éléments interactifs tels que les liens et les boutons reçoivent un **focus visuel clair** lorsqu'ils sont **sélectionnés à l'aide de la touche Tab du clavier**. Cela aide les utilisateurs de lecteurs d'écran à comprendre où se trouve le focus.
Exemple : Ajoutez une bordure ou un changement de couleur pour indiquer le focus.

 lun. 12/06

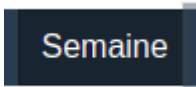
: Sans focus visuel donc actuellement non sélectionné via la navigation au clavier.

 mar. 13/06

: Avec un focus visuel donc actuellement sélectionné via la navigation au clavier.

 Jour

: Avec un focus visuel donc actuellement sélectionné via la navigation au clavier.

 Semaine

: Sans focus visuel mais couleur plus foncé donc l'élément suivant est actif et à subir un changement d'état après avoir cliquer dessus.

6. **Ordre de lecture et navigation au clavier** : Assurez-vous que le contenu est présenté dans un **ordre logique** pour une lecture cohérente par les technologies d'assistance. Veillez également à ce que tous les éléments interactifs, tels que les liens et les boutons, puissent être **accessibles et utilisables au clavier**. Testez la navigation au clavier pour vous assurer que les utilisateurs peuvent accéder à tous les éléments interactifs sans obstacle.

Exemple : Le **tabindex** est un attribut HTML qui détermine l'ordre de tabulation des éléments interactifs sur une page web. Il spécifie si un élément peut être ciblé et activé à l'aide de la touche Tab du clavier.

- Les éléments avec une valeur de tabindex positive sont **tabulables dans l'ordre croissant de leur valeur**. Par exemple, un élément avec `tabindex="1"` sera tabulé avant un élément avec `tabindex="2"`.
- La valeur zéro (**`tabindex="0"`**) : Cela signifie que l'élément **fait partie de l'ordre de tabulation normal de la page**. Il sera tabulé après tous les éléments avec une valeur de tabindex positive, dans l'ordre où ils apparaissent dans le code HTML.
- L'attribut absent ou la valeur négative (**`tabindex="-1"`**) : Cela indique que **l'élément ne peut pas être tabulé** à l'aide de la touche Tab, mais peut être

programmé pour recevoir le focus via JavaScript. Cela permet de rendre un élément accessible au clavier sans qu'il soit tabulable.

7. **Fournissez des transcriptions et des sous-titres pour les médias** : Pour les vidéos et les fichiers audio, assurez-vous de fournir des **transcriptions** et des **sous-titres** pour aider les utilisateurs malentendants ou qui ne peuvent pas accéder aux médias directement.

Exemple :

```
<video controls>
  <source src="video.mp4" type="video/mp4">
  <!-- Transcription pour les utilisateurs qui ne peuvent pas accéder à la vidéo -->
  <track src="transcription.vtt" kind="subtitles" srclang="fr" label="Français">
</video>
```

Utilisation du composant <v-video> de Vuetify :

```
<v-video src="chemin_vers_la_video.mp4">
  <track kind="subtitles" src="chemin_vers_les_sous-titres.vtt" srclang="fr"
label="French"></track>
</v-video>
```

8. **Autoriser et ajuster le zoom et la responsivité de la page** : Assurez-vous que votre application est compatible avec le zoom de la page. Les utilisateurs ayant une vision réduite doivent pouvoir **agrandir le contenu** sans perte de fonctionnalité ni de mise en page.

Exemple : Ajoutez les métadonnées de la vue dans le fichier index.html de votre application Vue.js.

```
<meta name="viewport" content="width=device-width, initial-scale=1.0,
maximum-scale=5.0">
```

Cela permettra au navigateur **d'ajuster le zoom** de la page en fonction des besoins de l'utilisateur.

Assurez-vous également que les composants que vous utilisez sont configurés pour **s'adapter automatiquement à la taille de l'écran**.

Exemple : Vous pouvez utiliser les classes responsives de Vuetify pour rendre vos composants réactifs aux différentes tailles d'écran :

```
<v-card class="mx-auto" max-width="600" width="100%">
```

```
<!-- Contenu de votre carte -->  
</v-card>
```

En utilisant la classe mx-auto, la carte sera centrée horizontalement, et les propriétés max-width et width définissent la taille maximale et la largeur de la carte. Ces valeurs peuvent être adaptées en fonction de vos besoins.

9. **Utiliser des attributs ARIA et des rôles** : Les attributs ARIA fournissent des **informations supplémentaires** sur les éléments HTML **pour les technologies d'assistance** telles que les lecteurs d'écran. Les attributs ARIA permettent également de **décrire l'état et les propriétés dynamiques d'un élément**. Par la suite, les **roles** permettent de **décrire le type d'élément et son rôle** dans l'interface.

Dans certaines situations, le choix des attributs ARIA et des rôles peut varier en fonction du contexte et des besoins spécifiques de l'interface utilisateur.

Exemples :

- aria-label : Utilisé pour fournir une étiquette alternative concise pour un élément qui n'a pas de texte visible.
- aria-labelledby : Utilisé pour faire référence à un élément visible (par son identifiant) qui sert de libellé pour un autre élément.
- aria-describedby : Utilisé pour faire référence à un élément visible (par son identifiant) qui décrit l'élément actuel en détail.

Boutons :

- role="button" : Utilisé pour indiquer qu'un élément est un bouton interactif.
- aria-label="texte explicatif" : Utilisé pour fournir une étiquette explicite au bouton lorsque le texte visible n'est pas suffisamment descriptif.
- aria-pressed="true/false" : Utilisé pour indiquer l'état pressé (activé/désactivé) d'un bouton bascule (toggle button).

- aria-expanded="true/false" : Utilisé pour indiquer l'état déplié/étendu d'un bouton d'expansion (expand button).

Formulaires :

- aria-label="texte explicatif" : Utilisé pour fournir une étiquette explicite à un élément de formulaire lorsque le texte visible n'est pas suffisamment descriptif.
- aria-required="true" : Utilisé pour indiquer qu'un champ de formulaire est obligatoire.
- aria-invalid="true" : Utilisé pour indiquer qu'un champ de formulaire contient une valeur invalide ou non conforme.
- role="form" : Utilisé est utilisé pour indiquer qu'un élément de formulaire

Menus :

- role="menu" : Utilisé pour indiquer qu'un élément est un menu.
- role="menuitem" : Utilisé pour indiquer qu'un élément est un élément de menu individuel.
- aria-haspopup="true" : Utilisé pour indiquer qu'un élément de menu a un sous-menu déroulant.

Onglets :

- role="tablist" : Utilisé pour indiquer qu'un élément contient une liste d'onglets.
- role="tab" : Utilisé pour indiquer qu'un élément est un onglet individuel dans une liste d'onglets.
- aria-selected="true/false" : Utilisé pour indiquer si un onglet est actuellement sélectionné.

Accordéons :

- role="tablist" : Utilisé pour indiquer qu'un élément contient une liste d'éléments d'accordéon.
- role="tab" : Utilisé pour indiquer qu'un élément est un élément d'accordéon individuel.
- role="tabpanel" : Utilisé pour indiquer le contenu associé à un élément d'accordéon.

Liens :

- aria-label="texte explicatif" : Utilisé pour fournir une étiquette explicite à un lien lorsque le texte visible n'est pas suffisamment descriptif.
- aria-current="page" : Utilisé pour indiquer qu'un lien représente la page ou l'emplacement actuel.
- role="link" : Utilisé pour indiquer qu'un élément est un lien hypertexte cliquable.

Images :

- aria-describedby="idDescription" : Utilisé pour associer une description plus détaillée à une image en référençant l'ID d'un élément de description.
- role="img" : Utilisé pour indiquer qu'un élément est une image (par exemple, pour les icônes).
- role="presentation" : Utilisé pour indiquer qu'une image est purement décorative et n'a pas de signification contextuelle. Cela permet aux lecteurs d'écran de sauter l'image sans en fournir de description.
- role="link" : Utilisé pour indiquer qu'une image fonctionne comme un lien cliquable, vous pouvez utiliser le rôle "link" pour indiquer sa fonctionnalité.

Tableaux :

- role="table" : Utilisé pour indiquer qu'un élément est un tableau.
- role="row" : Utilisé pour indiquer qu'un élément est une ligne dans un tableau.
- role="columnheader" : Utilisé pour indiquer qu'un élément est l'en-tête d'une colonne.
- role="rowheader" : Utilisé pour indiquer qu'un élément est l'en-tête d'une ligne.

Carrousels :

- role="list" : Utilisé pour indiquer qu'un élément est une liste de diapositives.
- role="listitem" : Utilisé pour indiquer qu'un élément est une diapositive individuelle dans un carrousel.
- aria-roledescription="diapositive" : Utilisé pour décrire le rôle de chaque élément de diapositive.

Messages d'erreur ou de confirmation :

- role="alert" : Utilisé pour indiquer qu'un élément est un message d'alerte important pour l'utilisateur.

- role="status" : Utilisé pour indiquer qu'un élément fournit des informations de statut ou de confirmation.

Groupes d'éléments :

- role="group" : Utilisé pour regrouper un ensemble d'éléments connexes, tels que des boutons ou des cases à cocher, afin de les identifier comme un groupe logique.

- aria-labelledby="idTitre" : Utilisé pour associer un élément de titre à un groupe d'éléments afin de fournir une description ou une étiquette pour le groupe.

Zones de navigation :

- role="navigation" : Utilisé pour indiquer qu'une zone de contenu contient des liens de navigation principaux.

- aria-label="texte explicatif" : Utilisé pour fournir une étiquette explicite à une zone de navigation lorsque le texte visible n'est pas suffisamment descriptif.

Menus contextuels :

- role="menu" : Utilisé pour indiquer qu'un élément est un menu contextuel.

- role="menuitem" : Utilisé pour indiquer qu'un élément est un élément de menu individuel dans un menu contextuel.

Contrôles personnalisés :

- role="slider" : Utilisé pour indiquer qu'un élément est un contrôle de curseur.

- role="progressbar" : Utilisé pour indiquer qu'un élément représente une barre de progression.

- role="radio" : Utilisé pour indiquer qu'un élément est un bouton radio.

- role="checkbox" : Utilisé pour indiquer qu'un élément est une case à cocher.

Zones de mise en évidence ou de focus :

- role="status" : Utilisé pour indiquer qu'un élément fournit des informations de statut ou de confirmation.

- role="alert" : Utilisé pour indiquer qu'un élément est un message d'alerte important pour l'utilisateur.

- role="tooltip" : Utilisé pour indiquer qu'un élément contient une information contextuelle qui apparaît au survol ou au focus.

Utilisation de l'état et des propriétés ARIA :

- aria-disabled : Utilisé pour indiquer qu'un élément interactif est désactivé.
- aria-checked : Utilisé pour indiquer si un élément de sélection (comme une case à cocher) est coché ou non.
- aria-expanded : Utilisé pour indiquer si un élément (comme un accordéon) est étendu ou réduit.
- aria-hidden : Utilisé pour indiquer qu'un élément doit être ignoré par les technologies d'assistance. Cela peut être utile pour masquer des éléments décoratifs ou redondants.

10. Documentez l'accessibilité : Fournissez une **documentation décrivant les fonctionnalités d'accessibilité** de votre application. Cela aide les utilisateurs à comprendre comment utiliser les fonctionnalités d'accessibilité et à bénéficier pleinement de votre application.

11. Testez : Premièrement, avec **des outils de tests d'accessibilité** ou avec **des lecteurs d'écran**. Cela vous aidera à identifier les problèmes potentiels et à les résoudre. Effectuez ensuite des tests sur **différentes plateformes et navigateurs**. Assurez-vous que votre application est testée sur différentes plateformes (ordinateurs de bureau, appareils mobiles, tablettes) et dans différents navigateurs (Chrome, Firefox, Safari, Edge, etc.). Enfin, faites également des tests avec de **vrais utilisateurs**. Les tests avec de vrais utilisateurs ayant des besoins en matière d'accessibilité peuvent fournir des **informations précieuses sur les problèmes d'accessibilité** de votre application ou de votre site internet.

Voici quelques exemples d'outils de test d'accessibilité et de lecteurs d'écran couramment utilisés :

Lighthouse (Chrome) : Lighthouse est un outil d'audit intégré dans les outils de développement de Chrome. Il fournit une analyse détaillée de la performance, de l'accessibilité, de l'optimisation pour les moteurs de recherche (SEO) et des bonnes pratiques générales.

Installation et utilisation :

<https://chrome.google.com/webstore/detail/lighthouse/blipmdconlkpinefehnmjammfjpmpbjk?hl=fr> et taper Lighthouse ou Outils de développement de Chrome -> Onglet "Audits" -> Sélectionner "Accessibility" -> Exécuter l'audit.

Axe DevTools (Chrome, Firefox) : Axe DevTools est une extension de navigateur qui peut être utilisée avec Chrome DevTools ou Firefox Developer Tools. Elle fournit des informations détaillées sur les problèmes d'accessibilité dans votre application web.

Installation et utilisation :

Chrome : Installer l'extension Axe DevTools dans le navigateur en recherchant sur <https://chrome.google.com/webstore/detail/lighthouse/blipmdconlkpinefehnmjammfjpmpbjk?hl=fr> -> Ouvrir les outils de développement -> Onglet "Axe".

Firefox : <https://addons.mozilla.org/> et recherchez "Axe DevTools" et suivez les instructions pour l'installer. -> Ouvrez les outils de développement de votre navigateur (F12 ou Ctrl+Shift+I) -> Recherchez l'onglet ou l'option "Axe" pour exécuter l'audit d'accessibilité.

Wave (Chrome, Firefox) : Wave est une extension de navigateur qui permet de détecter les problèmes d'accessibilité dans les pages web. Elle fournit des indications visuelles pour identifier les erreurs et les avertissements d'accessibilité.

Installation et utilisation :

Chrome : Installer l'extension Wave dans le navigateur en recherchant sur <https://chrome.google.com/webstore/detail/lighthouse/blipmdconlkpinefehnmjammfjpmpbjk?hl=fr> -> Cliquer sur l'icône Wave pour lancer l'analyse.

Firefox : <https://addons.mozilla.org/> et recherchez "Wave" et suivez les instructions pour l'installer dans votre navigateur -> Cliquez sur l'icône Wave pour lancer l'analyse d'accessibilité de votre page.

NVDA (Windows) : NVDA (NonVisual Desktop Access) est un lecteur d'écran open source populaire pour Windows. Il permet de tester l'accessibilité d'une application web en simulant l'expérience des utilisateurs ayant une déficience visuelle.

Installation et utilisation : <https://www.nvaccess.org/>

VoiceOver (Mac) : VoiceOver est le lecteur d'écran intégré sur les appareils Mac. Il peut être utilisé pour tester l'accessibilité des applications web sur macOS. Activez VoiceOver dans les préférences d'accessibilité de votre Mac, puis utilisez-le pour naviguer et interagir avec votre application web.

Installation et utilisation : Activer VoiceOver dans les préférences système -> Sélectionnez "Accessibilité" -> Dans l'onglet "Vision", activez VoiceOver.

AlInspector Sidebar (Firefox) : AlInspector Sidebar est une extension de Firefox qui permet de vérifier l'accessibilité des pages web. Elle fournit des rapports détaillés sur les problèmes d'accessibilité détectés.

Installation et utilisation : <https://addons.mozilla.org/> et recherchez "AlInspector Sidebar" et suivez les instructions pour l'installer -> Cliquez sur l'icône AlInspector Sidebar dans la barre d'outils pour lancer l'outil d'audit d'accessibilité sur votre page.

Orca (Linux) : Orca est un lecteur d'écran open source pour Linux, qui offre une expérience similaire à NVDA sur Windows et VoiceOver sur Mac. Il peut être utilisé pour tester l'accessibilité des applications web sur les distributions Linux. Orca est généralement inclus dans les environnements de bureau GNOME. Vous pouvez l'activer et le configurer via les paramètres d'accessibilité de votre système Linux.

Installation et utilisation : Vérifiez si Orca est déjà installé sur votre système en le recherchant dans les menus ou en exécutant la commande 'orca' ou 'orca --setup' dans un terminal. Sinon, vous pouvez l'installer à partir du gestionnaire de paquets de votre distribution Linux. Par exemple, pour Ubuntu, vous pouvez exécuter la commande 'sudo apt install gnome-orca' dans un terminal pour installer Orca.

JAWS (Windows et PAYANT) : Lecteur d'écran populaire pour les utilisateurs ayant une déficience visuelle sur les systèmes Windows. Il fournit une assistance auditive en convertissant le contenu textuel affiché à l'écran en discours ou en braille, permettant ainsi aux utilisateurs d'accéder aux informations et d'interagir avec les applications.

Installation et utilisation : <https://www.freedomscientific.com/>

"Contrast Checker" de WebAIM : Outil en ligne gratuit qui permet de vérifier le contraste entre deux couleurs et de déterminer s'ils respectent les critères d'accessibilité définis par les WCAG (Web Content Accessibility Guidelines).

Utilisation : <https://webaim.org/resources/contrastchecker/>

- Sélectionner les couleurs : Dans la section "Contrast Checker", vous verrez deux champs de couleur, "Couleur d'arrière-plan" et "Couleur du texte". Cliquez sur les champs respectifs pour ouvrir un sélecteur de couleurs.
- Vérifier le contraste : Une fois que vous avez sélectionné les deux couleurs, cliquez sur le bouton "Vérifier le contraste". L'outil affichera immédiatement les résultats. Les résultats indiquent si le contraste entre les deux couleurs est suffisant ou non, en fonction des recommandations d'accessibilité.

Pa11y (en ligne de commande) : Pa11y est une suite d'outils d'audit d'accessibilité en ligne de commande. Il permet d'automatiser les tests d'accessibilité et de générer des rapports détaillés sur les problèmes détectés.

Installation et utilisation : <https://pa11y.org/>

Exemple sur Linux :

1. Installez Node.js et npm (Node Package Manager) :

Vérifiez si Node.js est déjà installé en exécutant la commande suivante :

```
node -v
```

Les commandes varient en fonction de votre distribution Linux :

- a. Pour Ubuntu ou Debian :

```
sudo apt update
```

```
sudo apt install nodejs npm
```
- b. Pour CentOS, Fedora ou RHEL :

```
sudo dnf install nodejs npm
```
- c. Pour Arch Linux ou Manjaro :

```
sudo pacman -S nodejs npm
```

Une fois Node.js installé, vous pouvez vérifier si npm est également installé en exécutant la commande suivante :

```
npm -v
```

2. Ouvrez un terminal et exécutez la commande suivante pour installer Pa11y globalement :

```
npm install -g pa11y
```

3. Configuration : Créez un fichier de configuration pa11y.json pour définir les paramètres de Pa11y. Vous pouvez spécifier l'URL de la page à tester, les options de test et les rapports à générer. Voici un exemple de configuration :

```
{  
  "urls": ["https://example.com"],  
  "standard": "WCAG2AA",  
  "reporter": "html",  
  "reporterOptions": {  
    "file": "report.html"  
  }  
}
```

Dans cet exemple, nous testons l'URL `https://example.com` en utilisant la norme WCAG 2.0 niveau AA et générons un rapport au format HTML.

4. Exécution : lancer le test avec Pa11y : `~$ pa11y --config pa11y.json`
5. Analyse : Pa11y commencera à analyser la page spécifiée et affichera les résultats dans le terminal. Il affichera les problèmes d'accessibilité détectés, y compris leur description, le code HTML concerné et les recommandations pour les résoudre.
6. Rapport : Selon la configuration éditée, Pa11y générera également un rapport détaillé dans le format spécifié. Dans notre exemple, un rapport HTML sera généré dans un fichier nommé `report.html`.

III. Ressources et documentations

Voici une liste de ressources et de documentations sur l'accessibilité web que vous pouvez consulter pour approfondir vos connaissances :

1. Web Content Accessibility Guidelines (WCAG) : Les WCAG sont des recommandations internationales pour rendre les contenus web plus accessibles. Vous pouvez consulter la documentation officielle sur le site du W3C :

<https://www.w3.org/WAI/standards-guidelines/wcag/>

2. Documentation de Mozilla Developer Network (MDN) sur les rôles ARIA (Accessible Rich Internet Applications). :

<https://developer.mozilla.org/en-US/docs/Web/Accessibility/ARIA/Roles/>

Documentation de Mozilla Developer Network (MDN) sur les attributs ARIA (Accessible Rich Internet Applications). :

<https://developer.mozilla.org/en-US/docs/Web/Accessibility/ARIA/Attributes>

3. Mozilla Developer Network (MDN) - Accessibilité : MDN propose une section dédiée à l'accessibilité web, avec des guides, des tutoriels et des références pour différents aspects de l'accessibilité. Vous pouvez consulter leur documentation ici :

<https://developer.mozilla.org/fr/docs/accessibility>

https://developer.mozilla.org/fr/docs/Learn/Tools_and_testing/Cross_browser_testing/Accessibility

4. Inclusive Components : Inclusive Components est une collection d'exemples de composants web accessibles et inclusifs, accompagnés d'explications détaillées. Cela peut vous aider à comprendre comment implémenter des fonctionnalités accessibles dans vos projets. Vous pouvez consulter leur documentation ici :

<https://inclusive-components.design/>

5. Apple - Web Content Accessibility Guidelines (WCAG) Overview : Apple propose une vue d'ensemble des WCAG avec des conseils pratiques pour concevoir des expériences web accessibles sur les appareils Apple. Vous pouvez lire cette documentation ici :

<https://developer.apple.com/accessibility/guides/web-content-accessibility-overview/>

6. Documentation officielle de Vuetify, le framework de composants Vue.js sur les fonctionnalités d'accessibilité : <https://vuetifyjs.com/en/features/accessibility/>