

Évaluer l'accessibilité comme chef de projet non technique

Sommaire

1.	But de ce mémoire	3
2.	Mise en contexte	3
3.	Avant de commencer.....	4
4.	Poser les bases.....	4
5.	Former et outiller	5
5.1.	En interne	5
5.2.	Formations externes.....	6
5.3.	Accompagnement par un expert.....	6
5.4.	Outils du quotidien.....	6
6.	Présentation des outils de tests automatiques	7
6.1.	Wave Toolbar	7
6.2.	aXe Core.....	9
6.3.	Lighthouse de Google Chrome	10
6.4.	HeadingsMap.....	12
6.5.	Tanaguru contrast finder	13
6.6.	Valideur W3C.....	14
7.	Tester soi-même	14
7.1.	Tester le titre de la page	14
7.2.	Navigation au clavier	15
7.3.	Zoomer le texte	16
7.4.	Supprimer le CSS	16
7.5.	Tester avec un lecteur d'écran.....	17
8.	Quels critères sont couverts ?	17
9.	Restitution aux équipes.....	18
10.	Et après ?	18
11.	Conclusion	19

1. But de ce mémoire

L'accessibilité est un sujet riche pour lequel il existe plusieurs normes et référentiels (Web Content Accessibility Guidelines – WCAG – Référentiel Général d'Accessibilité pour les Administrations – RGAA – Accessiweb) et il n'est pas facile de s'y retrouver quand on entre dans ce monde pour la première fois.

Je me souviens de m'être longtemps posé la question au début de savoir quelle méthode utiliser entre RGAA et Accessiweb, tant leurs critères d'évaluation étaient identiques tout comme leur classification par niveaux (A, AA, AAA) ou encore si je devais utiliser la norme internationale WCAG.

Il y a beaucoup de critères à respecter et cela peut paraître complexe de prime abord. Mais, prise en compte dès le début, l'accessibilité est plus facile à mettre en place. Un certain nombre de critères peuvent être spécifiés en amont de la réalisation pour faciliter leur mise en œuvre par la suite.

Par ailleurs, il existe plusieurs outils de tests automatiques qui peuvent faciliter la tâche et détecter un certain nombre de points pouvant se révéler bloquants pour les personnes en situation de handicap ou utilisant des technologies d'assistance.

Dans ce mémoire, je vais montrer comment poser les bases d'un projet accessible et comment mener une base d'évaluation efficace sans avoir un profil technique et sans avoir à mettre les mains dans le code.

Une partie « gestion de projet » est abordée pour donner quelques pistes pour anticiper au mieux l'accessibilité mais le focus principal va être sur la méthode d'évaluation.

Le but est de montrer comment intégrer et tester l'accessibilité dans toutes les phases d'un projet et de permettre à un chef de projet sans profil technique d'avoir une idée du niveau d'accessibilité d'un site en estimant le nombre de critères couverts et lui donner les clés pour restituer les corrections à effectuer par son équipe (designers, développeurs, rédacteurs).

2. Mise en contexte

Un client demande que son site web « soit conforme aux normes d'accessibilité en vigueur ».

Sans plus de précisions ni compétences techniques, sans expérience de prise en compte de l'accessibilité dans un projet, comment répondre à ce besoin spécifique ?

3. Avant de commencer

Il faut anticiper la mise en œuvre de l'accessibilité dès la phase d'avant-vente. C'est un coût à considérer même si, intégrée dès le départ, l'accessibilité n'est pas un poste de dépense trop important. Selon si l'équipe intervenante est formée ou non, il faudra prévoir plus de temps, notamment en développement. La vélocité ne sera pas la même que d'habitude si l'équipe n'a jamais travaillé sur un tel projet. Selon le contexte, il sera possible de ventiler le coût de mise en œuvre de l'accessibilité entre tous les postes (augmenter le nombre de jours, le taux journalier) ou d'avoir une ligne « Accessibilité » dans le budget, cela peut être utile pour un éventuel financement.

Il faut aussi vérifier l'impact en termes de planning, est-ce que l'équipe va pouvoir produire les livrables dans les temps, sans empiéter sur les prochains projets ?

4. Poser les bases

Vouloir un site accessible c'est bien mais il faut poser les bonnes bases dès le départ pour éviter de se retrouver dans une situation où on passe de « je veux un site accessible » à « faisons au mieux et on verra plus tard pour atteindre le niveau pour être dans la norme requise ».

Des bonnes bases sont des bonnes spécifications. Quelques recherches mènent vite vers le RGAA.

Il est conseillé d'établir un Plan d'Assurance Qualité¹ – PAQ – et d'inscrire l'utilisation de ce référentiel comme base de travail ainsi que le niveau à atteindre (AA). Dans ce document, il faut noter qui interviendra dans le projet et comment sera assurée la bonne mise en œuvre de l'accessibilité. Cela va passer par des phases de tests, il faut en définir la fréquence (avant la livraison des wireframes/maquettes/front/back), qui les effectue et comment et quels outils vont être utilisés pour vérifier la conformité des livrables (outils d'audit automatique, audits par un expert en accessibilité, audits internes avec le RGAA).

En cas de non-conformités, quelle va être la procédure pour les corriger ? Cela peut être à l'aide d'une plateforme de tickets mise à disposition, en priorisant les bugs selon différentes importances. Qui va être responsable de la gestion de ces tickets, qui va les traiter et jusqu'à quand ? Par exemple, en cas de date de livraison impossible à repousser, il faut prévoir un niveau d'acceptation de livraison avec une maintenance corrective à postériori.

Il est recommandé de joindre une annexe contenant les points à respecter par chaque acteur du projet en les répartissant par métiers. Ainsi, chacun va pouvoir se

¹ https://fr.wikipedia.org/wiki/Plan_qualit%C3%A9

concentrer sur les critères qui le concernent. Cela va montrer également au client comment les tâches vont être découpées et réparties dans l'équipe.

Si l'organisation est en mode agile, le respect du niveau d'accessibilité peut être ajouté dans la définition de « fait ».

Au niveau des spécifications fonctionnelles et techniques, il est important de détailler le comportement attendu de certains éléments. Par exemple que la navigation clavier doit se faire dans l'ordre de lecture, comment gérer les alternatives aux images², comment gérer les textes masqués mais qui doivent être conservés pour les utilisateurs de lecteurs d'écran³, quels champs doivent être obligatoires dans le back-office (titres par exemple) si le contexte s'y prête, que les tableaux doivent avoir un titre et éventuellement un résumé, qu'il faut regrouper les champs de même nature, comment gérer les changements de langues. Si le site comporte des vidéos, comment les intégrer de manière accessible, de même que pour les composants dynamiques (carrousels, onglets, accordéons, etc.)⁴⁵.

Il sera nécessaire de s'entourer de personnes ayant des compétences spécifiques (ergonome, directeur technique) pour aider à rédiger ces spécifications.

5. Former et outiller

5.1. En interne

En partant de l'annexe précédemment créée, une présentation des critères à respecter peut être faite au cours d'une réunion.

Même si c'est une présentation en surface, elle est primordiale car chaque personne impliquée pourra creuser les critères qui la concernent par la suite.

Il est important que les designers soient au fait des critères qu'ils doivent respecter. Par exemple, on peut facilement imaginer un design présenté et validé par le client qui se révélerait non conforme en termes de contrastes. Selon le client, il pourrait être délicat de revenir dessus sans qu'une frustration et une perte de confiance ne s'installent.

Pour les développeurs, une des thématiques à respecter concerne les scripts, ce qui est une partie délicate à appréhender. Avec une présentation, ils pourront prendre le temps de rechercher et de tester des plugins accessibles qui répondront aux besoins du projet.

² <https://bitsofco.de/alternative-text-and-images/>

³ <https://www.ffoodd.fr/cache-cache-css/>

⁴ <https://dequeuniversity.com/library/>

⁵ <https://w3c.github.io/aria-practices/>

5.2. Formations externes

Il existe plusieurs formations spécifiques pour les développeurs, les chefs de projet ou les designers. Elles se déroulent le plus souvent sur 2 ou 3 jours. Une formation peut être prévue le plus rapidement possible pour que l'équipe soit opérationnelle au bon moment. Il faudra aussi former les contributeurs car un certain nombre de critères sont à respecter lors de la production de contenus. Là aussi, il existe des formations spécifiques, le plus souvent sur une seule journée.

5.3. Accompagnement par un expert

Un accompagnement par un expert peut être envisagé tout au long du projet. C'est même le moyen le plus sûr pour respecter l'accessibilité lors d'un premier projet. Dans ce cas, il faut définir avec lui s'il doit intervenir ponctuellement (pour valider les livrables par exemple) ou être disponible régulièrement pour répondre aux questions et guider les équipes en plus de tester les livrables.

5.4. Outils du quotidien

Il est important que chacun puisse tester ses livrables. Ainsi, il faut fournir les bons outils aux bonnes personnes.

Pour les designers, il existe des plugins pour Sketch⁶ ou Photoshop⁷ qui permettent de vérifier les ratios de contrastes. La notice d'Atalan pour les designers⁸ est aussi une excellente ressource ainsi que le guide du concepteur⁹ du RGAA.

Pour les développeurs, il existe aussi des notices¹⁰¹¹ faites par Atalan, des guides¹²¹³ du RGAA et des outils que je vais présenter ci-après.

En parcourant le RGAA, on constate que de nombreux critères sont liés aux contenus, à leur pertinence, à leur bonne restitution par les technologies d'assistance. Les contributeurs ont aussi leur notice¹⁴¹⁵¹⁶ qui explique chaque point à respecter de manière détaillée pour que les contenus soient rédigés de manière accessible.

⁶ <https://github.com/getflourish/Sketch-Color-Contrast-Analyser>

⁷ <https://www.adobeexchange.com/creativecloud.details.12170.html>

⁸ <https://www.accede-web.com/notices/graphique/>

⁹ <https://disic.github.io/guide-concepteur/>

¹⁰ <https://www.accede-web.com/notices/html-css-javascript/>

¹¹ <https://www.accede-web.com/notices/interface-riche/>

¹² <https://github.com/DISIC/guide-developpeur>

¹³ <https://github.com/DISIC/guide-integrateur>

¹⁴ <https://www.accede-web.com/notices/editoriale/>

¹⁵ https://github.com/DISIC/guides-documents_bureautiques_accessible

¹⁶ http://disic.github.io/guide-contribuer_accessible/

6. Présentation des outils de tests automatiques

Il existe plusieurs outils de tests automatiques. Cela peut-être des extensions navigateurs, des outils à télécharger ou des plugins à installer comme évoqué ci-dessus.

Certains de ces outils permettent de faire des audits rapides pour avoir une idée de l'accessibilité d'un site, d'autres sont utiles en amont (plugin pour tester les contrastes dans Sketch ou Photoshop).

Voici une liste - non exhaustive - d'outils :

6.1. Wave Toolbar

Il s'agit d'une extension disponible sur Firefox et Chrome qui scanne la page et met en surbrillance différentes informations comme les erreurs, les avertissements, les éléments ARIA utilisés, les niveaux de titres... Elle est tout à fait indiquée pour un chef de projet non technique car les éléments relevés sont identifiables immédiatement à l'aide de différentes icônes qui apparaissent sur la page.

Il est également possible de désactiver les styles pour savoir si le contenu du site reste compréhensible sans eux et de tester le ratio de contraste entre la couleur du texte et celle du fond.

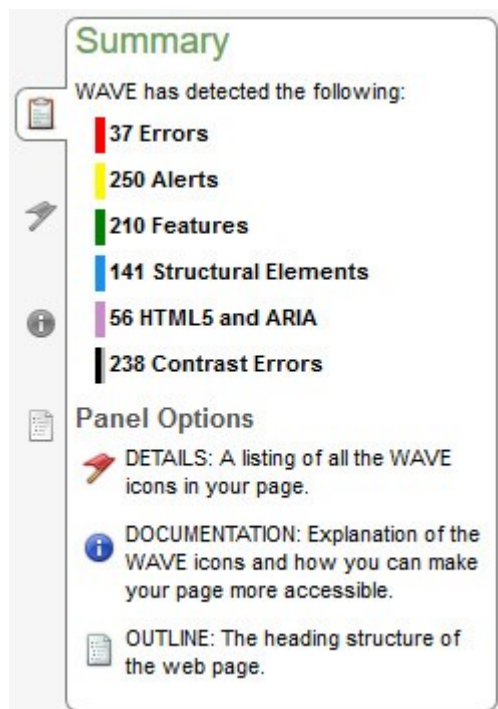


Figure 1 - Résultat d'une évaluation Wave

Un lien est disponible pour chaque information relevée. On peut voir d'un coup d'œil s'il s'agit d'une erreur, d'un avertissement, d'une information etc.

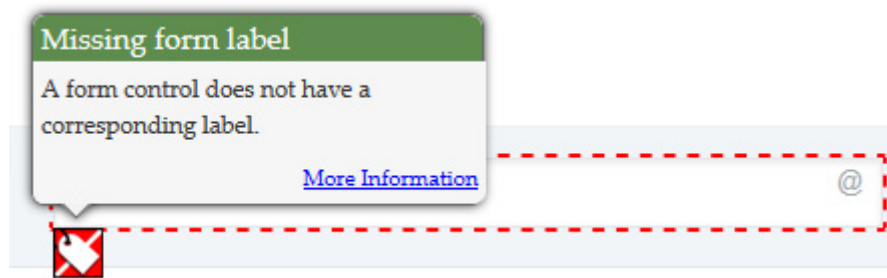


Figure 2 - Détail d'une erreur

Une description accompagne chacune de ces informations ainsi qu'un lien, permettant d'ouvrir un panneau de documentation.

Documentation

Errors

Missing form label

What It Means

Why It Matters

How to Fix It

The Algorithm... in English

Standards and Guidelines

Icon index

Errors
Missing form label

What It Means
A form control does not have a corresponding label.

Why It Matters
If a form control does not have a properly associated text label, the function or purpose of that form control may not be presented to screen reader users. Form labels also provide visible descriptions and larger clickable targets for form controls.

How to Fix It
If a text label for a form control is visible, use the <label> element to associate it with its respective form control. If there is no visible label, either provide an associated label, add a descriptive title attribute to the form control, or reference the label(s) using aria-labelledby. Labels are not required for image, submit, reset, button, or hidden form controls.

The Algorithm... in English
An <input> (except types of image, submit, reset, button, or hidden), <select>, or <textarea> does not have a properly associated label text. A properly associated label is:
a <label> element with a for attribute value that is equal to the id of a unique form control
a <label> element that surrounds the form control, does not surround any other form controls, and does not reference another element with its for attribute
a non-empty title attribute, or
a non-empty aria-labelledby attribute.

Standards and Guidelines

- [Section 508 \(n\)](#)
- [1.1.1 Non-text Content \(Level A\)](#)
- [1.3.1 Info and Relationships \(Level A\)](#)
- [2.4.6 Headings and Labels \(Level AA\)](#)
- [3.3.2 Labels or Instructions \(Level A\)](#)

[Icon index](#)

Figure 3 - Panneau d'explication de l'erreur

6.2. aXe Core

C'est un des tests automatiques les plus populaires¹⁷. Il couvre 67 règles (en mai 2018) correspondant aux WCAG 2.0, à la section 508 (spécifique aux États-Unis) et des bonnes pratiques de code (ex. Titre de niveau 1 présent dans la page). L'équipe de développement d'aXe se concentre sur la rapidité et la fiabilité des tests effectués. Dans leur manifeste, aXe doit retourner 0 faux-positif, être rapide et fonctionner sur les navigateurs les plus récents.

Lorsqu'une erreur est trouvée, un descriptif détaillé est mis à disposition et permet de savoir quel référentiel et quel critère sont en violation, à quelle catégorie elle appartient (couleur, navigation clavier...), où se trouve l'erreur et comment la corriger.

On peut aussi mettre la zone en erreur en surbrillance et l'atteindre directement dans le DOM.

C'est un outil principalement à destination des développeurs. Un chef de projet va pouvoir s'en servir pour faire remonter les grandes thématiques qui posent problème (les contrastes, les labels manquants...). Grâce à l'option de mise en surbrillance, il sera possible de donner des exemples de bugs plus détaillés en montrant la zone qui pose problème comme la capture ci-dessous.

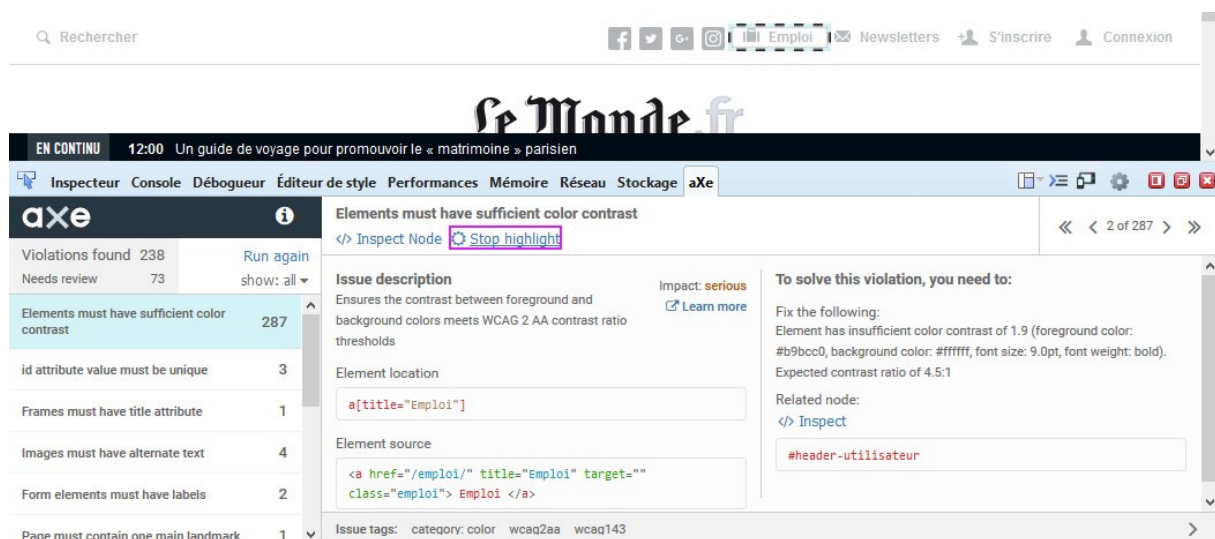


Figure 4 - Détail d'une erreur dans aXe. Ici, l'entrée de menu « emploi » est en surbrillance et n'a pas un ratio de contraste suffisant.

¹⁷ <https://www.deque.com/blog/accessibility-library-axe-core-1-million-downloads/>

6.3. Lighthouse de Google Chrome

Il utilise aXe Core¹⁸ mais liste en plus des tests à effectuer manuellement.

Il est possible de s'en servir comme aXe car il s'agit aussi d'un outil principalement destiné aux développeurs. La même utilisation pourra en être faite par un chef de projet, à savoir, lister les thématiques qui posent problème. Cet outil sera plus adapté qu'aXe s'il faut tester le niveau de performance web, de SEO et de Progressive Web App.

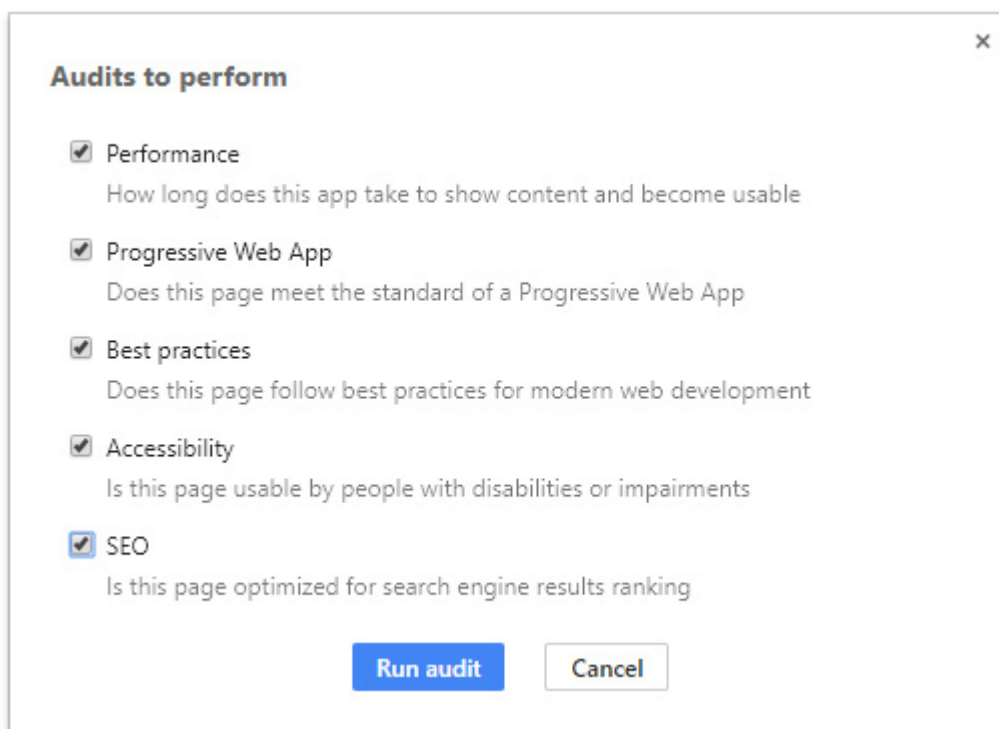


Figure 5 - Configuration de l'audit de Chrome

¹⁸ <https://www.deque.com/blog/google-selects-deques-axe-chrome-devtools/>

Le volet accessibilité liste les erreurs et comment les corriger. Il indique aussi les tests pouvant être fait manuellement, les tests réussis et ceux non applicables.

▶ Additional items to manually check

Accessibility

These checks highlight opportunities to [improve the accessibility of your web app](#). Only a subset of accessibility issues can be automatically detected so manual testing is also encouraged.

93

Color Contrast Is Satisfactory

These are opportunities to improve the legibility of your content.

▶ Background and foreground colors do not have a sufficient contrast ratio.

▶ 17 Passed Audits

▶ 17 Not Applicable Audits

▶ Additional items to manually check

These items address areas which an automated testing tool cannot cover. Learn more in our guide on [conducting an accessibility review](#).

▶ The page has a logical tab order

▶ Interactive controls are keyboard focusable

▶ The user's focus is directed to new content added to the page

▶ User focus is not accidentally trapped in a region

▶ Custom controls have associated labels

▶ Custom controls have ARIA roles

▶ Visual order on the page follows DOM order

Figure 6 - Panneau Accessibilité listant le score atteint, les tests réussis et non réussis et ceux à tester manuellement

6.4. HeadingsMap

Cette extension pour Firefox et Chrome permet en un coup d'œil de vérifier la structure des titres. Un ordre de titre cohérent est essentiel pour les personnes naviguant avec un lecteur d'écran car cela peut leur servir de lien d'évitement. Un ordre de titre incohérent peut fortement perturber leur navigation.

La structure des titres doit être <h1> <h2> <h3>... et non une structure avec des sauts de niveaux comme ainsi : <h1>, <h2>, <h4>, <h3>.

Attention cependant, HeadingsMap ne détecte pas si un titre a été créé à l'aide d'ARIA (grâce au rôle heading et à l'attribut aria-level). Pour tester la présence de ces titres, sans analyser le code, rien de mieux que d'utiliser un lecteur d'écran.

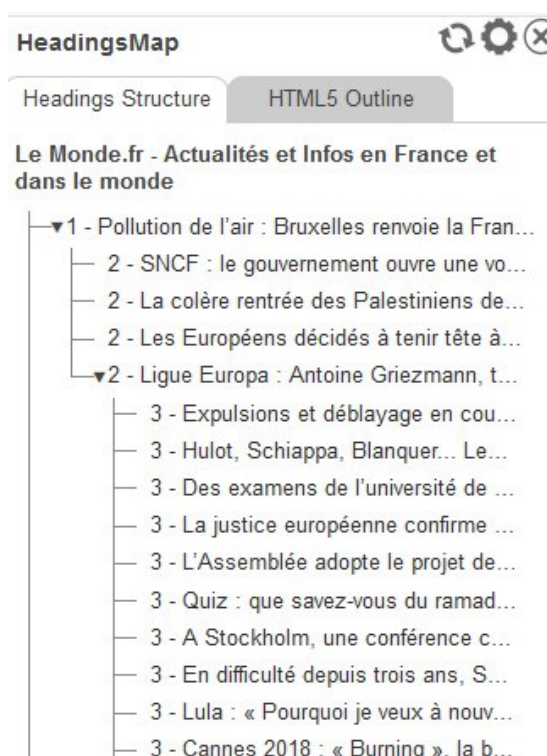


Figure 7 - Structure des titres sur le site lemonde.fr

6.5. Tanaguru contrast finder

Ce site permet de tester le ratio de contraste entre le texte et la couleur de fond et propose une palette de couleurs valides si le test initial n'est pas conforme. Il peut être utilisé en amont par un designer lorsqu'il crée ses maquettes ou par un chef de projet pour s'assurer que les ratios sont corrects en regard du niveau de conformité attendu.

Couleur du texte :

Pour chaque couleur (rouge, vert et bleu), indiquer un nombre entre 0 et 255.

Rouge : Vert : Bleu :

La couleur doit être comprise entre #000000 à #FFFFFF

Hexadécimal :

Couleur du fond :

Pour chaque couleur (rouge, vert et bleu), indiquer un nombre entre 0 et 255.

Rouge : Vert : Bleu :

La couleur doit être comprise entre #000000 à #FFFFFF

Hexadécimal :

Ratio minimum :

Dans la réglementation internationale établie par les [WCAG](#), le [critère de succès 1.4.3](#) demande pour le texte un rapport de contraste minimum de 4.5:1 (et 3:1 pour le texte agrandi).

Ce rapport de contraste minimum est également exigé par la réglementation française, établie par le [RGAA](#) 3.0 2016, dans les [critères 3.3 et 3.4](#).

Composante à modifier :

☒ Modifier la couleur du texte

☐ Modifier la couleur du fond

Propose-moi :

☒ les couleurs valides et *très proches* de la couleur initiale

☐ une *palette* de couleurs valides

Figure 8 - Configuration du test de contraste

Ancien contraste

Avant plan	Arrière plan	Exemple	Ratio	Distance
hsl(120, 32%, 40%) rgb(70, 136, 71) #468847	hsl(102, 44%, 89%) rgb(223, 240, 216) #DFF0D8	Titre de grande taille avec des mots en gras Ceci est un échantillon de texte avec quelques mots en gras pour illustrer le bon contraste.	3.61029	

Nouveau contraste : 22 résultats (1 786 080 couleurs testées)

Avant plan	Arrière plan	Exemple	Ratio	Distance
hsl(125, 15%, 38%) rgb(83, 114, 86) #537256	hsl(102, 44%, 89%) rgb(223, 240, 216) #DFF0D8	Titre de grande taille avec des mots en gras Ceci est un échantillon de texte avec quelques mots en gras pour illustrer le bon contraste.	4.50057	17.19
hsl(123, 28%, 36%) rgb(66, 118, 69) #427645	hsl(102, 44%, 89%) rgb(223, 240, 216) #DFF0D8	Titre de grande taille avec des mots en gras Ceci est un échantillon de texte avec quelques mots en gras pour illustrer le bon contraste.	4.50074	18.07
hsl(122, 32%, 35%) rgb(61, 119, 63) #3D773F	hsl(102, 44%, 89%) rgb(223, 240, 216) #DFF0D8	Titre de grande taille avec des mots en gras Ceci est un échantillon de texte avec quelques mots en gras pour illustrer le bon contraste.	4.50078	18.33

Figure 9 - Résultat du test avec une proposition de palette de couleurs valides

6.6. Valideur W3C

Un des critères de base est la validité du code HTML. Cela permet de s'assurer qu'il n'y aura pas d'effet de bord ou de mauvaise interprétation du contenu par les technologies d'assistance. Un rapide copier-coller de l'URL dans le valideur¹⁹ et les éventuelles erreurs sont remontées de manière détaillée. Un chef de projet va s'assurer ici qu'il n'y ait pas d'erreurs qui remontent. S'il y en a, il pourra simplement transmettre l'URL de résultat aux équipes pour qu'elles voient qui doit corriger quel point. Certaines erreurs peuvent venir du front et d'autres du back.

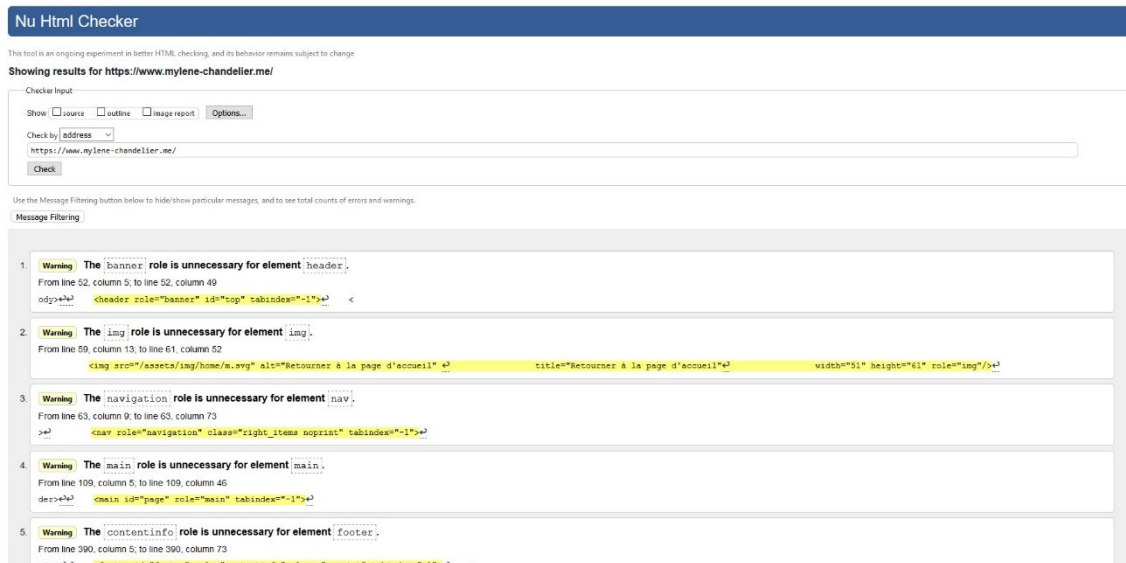


Figure 10 - Résultat du valideur w3C indiquant des avertissements

7. Tester soi-même

Quelques tests simples et rapides peuvent être effectués manuellement pour compléter les tests automatiques.

7.1. Tester le titre de la page

Il faut vérifier que le titre de la page (présent dans l'onglet du navigateur) est toujours cohérent. Il est préférable d'avoir d'abord le titre de la page puis le nom du site séparés par un caractère (« - » ou « | » le plus souvent).

Si le contenu de la page est modifié, par exemple en cas d'erreur dans un formulaire ou d'une requête de recherche et que la page est rafraîchie, le titre de la page doit être mis à jour en conséquence. Si ce n'est pas le cas, c'est une erreur qui peut être remontée à l'équipe de développement. Ces tests permettent de s'assurer qu'un utilisateur de lecteur d'écran saura toujours où il se trouve et si son action a bien été effectuée.

¹⁹ <https://validator.w3.org/>

7.2. Navigation au clavier

En utilisant la touche « tab » pour parcourir le site, l'ordre de navigation doit rester cohérent selon le sens de lecture. Tous les liens, champs de formulaires, boutons ou autres éléments interactifs doivent être atteignables au clavier et visibles à la prise de focus. Par exemple une bordure, un fond de couleur ou tout style suffisamment visible doivent permettre de distinguer ces zones.

7.2.1. Liens d'évitement

La première tabulation doit idéalement atterrir sur un lien d'évitement. Ce sont des liens placés au début du code pour permettre de naviguer rapidement d'une zone principale à une autre. Par exemple, « aller au contenu », « aller au pied de page ». Ces liens doivent être visibles à la prise de focus.

7.2.2. Fenêtres modales

Si des modales (ou popin) sont présentes, il faut que le focus se place sur le premier élément interactif qui se trouve à l'intérieur de celle-ci (bouton de fermeture le plus souvent) et qu'elle se ferme à l'aide de la touche échap. Enfin, il ne faut pas pouvoir continuer de naviguer dans la page lorsqu'une modale est ouverte. Le focus doit être bloqué à l'intérieur de celle-ci.

7.2.3. Formulaires

Une bonne implémentation de la navigation clavier dans un formulaire est faite si :

- On passe d'un champ à l'autre dans un ordre logique et que le focus sur le champ est visible
- Les cases à cocher et boutons radio peuvent être cocher/décocher via la barre espace
- Une liste de choix (select) s'ouvre avec la barre espace
- On peut soumettre le formulaire avec la touche entrée ou espace.

À noter que tous ces comportements sont présents nativement. Si ce n'est pas le cas, cela peut être parce qu'une librairie JavaScript a été utilisée pour styler certains composants (notamment les boutons radio, select et cases à cocher).

L'intérêt de ce test est de vérifier que l'utilisateur sait toujours où il se trouve et qu'il peut naviguer dans un ordre logique.

7.3. Zoomer le texte

L'augmentation du niveau de zoom doit rendre la lecture plus aisée pour les personnes malvoyante. Mais il arrive que le code soit trop rigide et ne permette pas qu'une zone s'adapte en fonction de l'agrandissement du texte.

Pour tester cela, il faut aller dans Affichage > Zoom > Zoom texte seulement dans Firefox.

Ensuite, zoomer jusqu'à 200 % et s'assurer que l'ensemble du contenu est visible et lisible.

Il ne doit pas y avoir de textes qui se superposent ou qui disparaissent à l'intérieur d'un bloc. Si c'est le cas, il s'agira d'un bug côté front, le plus souvent dû à une hauteur fixe ou un positionnement absolu.

La conception responsive aide à couvrir cette problématique assez facilement. En zoomant à 200%, il est possible de passer en vue « mobile ».


 Recherche Emploi Newsletters S'inscrire Connexion

Figure 11 - Le zoom à 200% provoque un masquage du texte à cause de la hauteur fixe du menu

 Recherche Emploi Newsletters S'inscrire Connexion

Figure 12 - Le même niveau de zoom en ayant remplacé la hauteur fixe dans le CSS par un min-height.

7.4. Supprimer le CSS

L'extension Wave permet de supprimer les styles de la page. Le critère à respecter est de s'assurer que tous les contenus restent compréhensibles et visibles lorsque les styles sont désactivés. Par exemple, des liens de réseaux sociaux qui ne sont visibles qu'avec une icône doivent laisser apparaître un texte lorsque le CSS est désactivé.



Figure 12 - Liens de réseaux sociaux avec des icônes simples sur le site arsla.org



Figure 14 - Les mêmes liens de réseaux sociaux avec le CSS désactivé.

7.5. Tester avec un lecteur d'écran

Tester la restitution par un lecteur d'écran peut aider à lever de nombreuses barrières. Il est possible de tester avec NVDA, par exemple, qui est gratuit et très utilisé (avec Jaws sur Windows et VoiceOver, présent nativement sur MacOS et iOS). Cela permet de se rendre compte si le contenu est compréhensible, si les alternatives textuelles sont cohérentes. Cela permet aussi de confirmer que l'ordre des niveaux de titres est correct. En effet, une limite de HeadingsMap ne sait pas détecter si un titre est présent lorsqu'il est créé avec ARIA. Un lecteur d'écran permet de l'interpréter correctement.

Pour aller plus loin, la Direction interministérielle du numérique et du système d'information et de communication de l'État – DINSIC – a mis à disposition un guide²⁰ pour installer et tester des composants riches avec les lecteurs d'écran. Atalan a également publié un article à ce sujet²¹.

En complément de ces documentations, deux démonstrations sont disponibles pour tester les niveaux de titres²² et les textes masqués et la présence d'alternatives aux images²³. Ces tests ont été effectués avec NVDA / Firefox et Jaws / IE11.

8. Quels critères sont couverts ?

Avec cette méthode d'organisation, chacun doit s'être assuré de couvrir tous les critères qui lui sont attribués. Selon le temps à disposition, il est possible de faire une passe avec aXe ou Lighthouse et quelques tests manuels pour vérifier que c'est bien le cas. S'il y a plus de temps, on peut parcourir le maximum de critères pour voir s'ils

²⁰ https://disic.github.io/guide-lecteurs_ecran/

²¹ <https://blog.atalan.fr/tester-accessibilite-web-pdf-lecteur-ecran-nvda/>

²² <https://www.mylene-chandelier.me/pres/ean/demo/titres.html>

²³ <https://www.mylene-chandelier.me/pres/ean/demo/textes-masques.html>

sont respectés et tester avec un lecteur d'écran. Dans un contexte WCAG, on pourrait aussi suivre cette liste de tests²⁴.

9. Restitution aux équipes

Selon le contexte, il peut être intéressant de prioriser la correction des bugs. Par exemple en fonction de la sévérité (répétition sur plusieurs pages), l'impact en termes de coût, l'impact utilisateur...

Avec des outils comme Trello, Jira ou Mantis, on peut faire un ticket par non-conformité relevée, en l'illustrant si besoin et en lui ajoutant un lien vers le critère du référentiel utilisé. Ce guide²⁵ peut aider à prioriser les retours par impact utilisateur.

10. Et après ?

L'accessibilité d'un site ne s'arrête pas à la mise en ligne, un site vit et, si les contributeurs ne sont pas vigilants à leur façon d'intégrer les contenus, il peut vite devenir non conforme. À chaque nouveau développement ou contribution, il faut être vigilant sur le respect de l'accessibilité pour maintenir un bon niveau de conformité.

Il ne faut pas hésiter à former les contributeurs et à leur remettre la notice Atalan ou RGAA pour les guider.

Il est aussi possible de faire appel à un expert pour effectuer un audit. C'est une occasion d'améliorer les connaissances de l'équipe sur l'accessibilité et de faire mieux pour les prochains projets. Attention toutefois, faire un audit à la fin du projet est risqué s'il s'agit d'un premier projet. L'idéal est de se faire accompagner tout au long du projet pour éviter des retours importants à la fin.

Dans l'hypothèse d'un objectif de labellisation, un audit serait effectué par un expert en accessibilité numérique sur un échantillon de pages représentatif. Ainsi, si l'équipe n'était pas certaine de la conformité de certains critères (celui des scripts étant le plus délicat à respecter), l'audit le remonterait et l'expert suggérerait des corrections. S'il était possible de faire les corrections, un nouvel audit serait effectué pour les vérifier et présenter le niveau d'accessibilité obtenu. Si des non-conformités étaient encore relevées et qu'il était possible de faire les corrections, un dernier audit serait effectué pour délivrer le label e-accessible. Pour être en conformité avec la loi²⁶, il faut atteindre le niveau 4 sur 5. Il est assez aisé de passer au niveau 5 car il est demandé de couvrir à minima un critère de niveau AAA. Par exemple, si le texte n'est jamais justifié sur le site ou qu'un fil d'ariane est présent, un de ces critères est respecté.

²⁴ <https://www.w3.org/WAI/test-evaluate/preliminary/>

²⁵ https://disic.github.io/guide-impacts_utilisateurs/

²⁶ <http://references.modernisation.gouv.fr/e-accessible>

Dans le contexte de la labellisation, une visite de contrôle est effectuée au bout de 18 mois (le label e-accessible étant valide 3 ans).

11. Conclusion

L'accessibilité n'est pas la cerise sur le gâteau. Elle fait partie du projet dès le départ, au même titre que le SEO ou la performance. Il est, de fait, indispensable de spécifier en amont les actions à mener par les différents acteurs du projet pour la respecter car il est très compliqué (et coûteux) de l'inclure à posteriori.

Concernant les tests automatiques, on estime qu'environ 30% des critères peuvent être validés de manière automatique²⁷.

C'est un excellent point de départ pour se faire une idée de l'accessibilité d'un site et corriger les erreurs les plus importantes. L'utilisation de ces outils se fait généralement de manière aisée et rapide.

Tant que l'on relève des erreurs avec des tests automatiques il est inutile de faire appel à un expert pour faire un audit complet.

Une passe manuelle est toujours nécessaire car un robot ne peut pas juger de la pertinence d'une alternative textuelle, vérifier l'ordre de tabulation ou voir si le zoom à 200% rend le contenu incompréhensible.

Si un nombre important d'erreurs est relevé trop facilement, c'est que la démarche de mise en conformité d'accessibilité n'a pas été bien comprise. Dans ce cas, il peut s'agir d'un manque de formation qui est vraiment indispensable pour un premier projet. Sans formation, on peut se faire accompagner par un expert tout au long du projet pour qu'il réponde aux questions des équipes, fasse des audits réguliers et suggère des corrections.

Il est important d'impliquer l'ensemble de l'équipe dans la démarche de mise en conformité d'accessibilité. En responsabilisant chacun à livrer des maquettes et développement accessibles, les tests se feront plus rapidement sur le court terme et les chances de livrer un projet avec un bon niveau d'accessibilité seront améliorées.

Mais surtout, tout le monde montera en compétence et certaines bonnes pratiques pourront s'instaurer d'elles-mêmes. Il y a un certain nombre de critères qui sont très simples à respecter et qui ne prennent pas beaucoup de temps à implémenter. Ces critères pourront être intégrés dans n'importe quel projet même s'il n'est pas prévu de respecter l'accessibilité.

²⁷ D'après une étude très complète effectuée par le gouvernement Britannique qui est régulièrement mise à jour, la dernière en date est d'avril 2018 <https://alphagov.github.io/accessibility-tool-audit/index.html>

Et qui sait, cela éveillera peut-être des vocations d'experts en accessibilité numérique 😊