paradigm
.brussels

RECOMMANDATIONS INCLUSIVES ET RÉFÉRENTIEL TECHNIQUE PERMETTANT D'AMÉLIORER ET D'OPTIMISER LES SERVICES PUBLICS NUMÉRISÉS POUR LES PERSONNES PRÉSENTANT UN RISQUE DE FRACTURE NUMÉRIQUE

_INCLUSION NUMERIQUE

TABLE DES MATIERES

I. Introduction

Paradigm a mandaté le CAWaB pour apporter des recommandations inclusives dans le cadre du développement des services publics numérisés (SPN) en Région de Bruxelles-Capitale.

Dans un premier rapport, a été développé une analyse approfondie de l'utilisation de deux SPN (MyActiris & IRISbox) par 12 publics présentant un risque de fracture numérique, répartis comme suit :

1) **4 personnes dyslexiques** ;
2) **4 personnes avec une déficience intellectuelle ;**
3) **6 personnes aveugles** : personnes aveugles utilisatrices d'une synthèse vocale ;
4) **2 personnes malvoyantes** : personnes malvoyantes utilisatrices d'une synthèse vocale et/ou de logiciels d'agrandissement/contraste ;
5) **7 personnes sourdes** : personnes qui ont une surdité profonde ;
6) **2 personnes malentendantes** : personnes porteuses d'un appareil auditif ;
7) **5 personnes utilisant le clavier** : personnes utilisant uniquement le clavier pour naviguer sur internet ;
8) **5 personnes âgées** : personnes de plus de 65 ans[1] qui n'ont pas d'ordinateur au sein de leur foyer ;
9) **4 personnes de langue étrangère** : personnes ne parlant et ne comprenant pas le français. Les langues étrangères choisies (en raison de la nécessaire présence d'interprètes) ont été l'espagnol et l'arabe ;
10) **4 personnes précarisées** : personnes avec peu de moyens financiers qui ont le diplôme du CEB (primaire) ;
11) **5 personnes en formation d'alphabétisation**: personnes qui ont des notions de lecture et d'écriture du français ;
12) **7 personnes sur smartphone**: personnes qui ne possèdent qu'un smartphone pour se connecter à internet y compris **2 personnes malvoyantes** utilisatrices d'une synthèse vocale.

Dans chaque groupe cible étaient présents au moins un chercheur d'emploi, un jeune de moins de 25 ans et une femme.

Pour toucher ces publics, il a été fait appel à un réseau de 23 associations.

Deux groupes contrôles ont aussi été constitués. Ceux-ci étaient composés de personnes qui ne sont pas considérées comme présentant un risque de fracture numérique, afin de pouvoir comparer leurs expériences avec les 12 groupes cibles présentés ci-dessus :

1) **6 personnes pour le groupe contrôle ordinateur**
2) **3 personnes pour le groupe contrôle smartphone**

Au total, 64 citoyens ont participé à l'expérience.

Cette étude a permis d'objectiver les besoins particuliers de certains publics lors de leur navigation sur les sites internet des services publics, mais également de tirer de nombreux enseignements plus généraux.

[1] Une exception a été faite à la découverte de l'âge d'une participante qui avait finalement 55 ans.

Un premier constat est sans appel : les normes d'accessibilité numérique, imposées par l'Ordonnance du 4 octobre 2018 relative à l'accessibilité des sites internet et des applications mobiles des organismes publics régionaux et des communes[2], même lorsqu'elles sont correctement appliquées - ce qui n'est malheureusement pas encore très souvent le cas- ne répondent pas aux besoins de l'ensemble de la population.

Ce rapport reprend donc 38 recommandations qui permettront de rendre, au-delà des normes d'accessibilité (WCAG 2.1 AA), les services publics en ligne inclusifs pour tous les citoyens.

En complément de ces recommandations, la deuxième partie de ce rapport présente un référentiel à destination des développeurs IT, répertoriant des critères techniques pour appliquer ces recommandations.

[2] Cette ordonnance transpose la Directive (UE) 2016/2102 du Parlement européen et du Conseil du 26 octobre 2016 relative à l'accessibilité des sites internet et des applications mobiles des organismes du secteur public.

II. Partie I : Recommandations pour des services publics numérisés inclusifs

A. Maintien d'alternatives aux canaux digitaux

Recommandation 1 : En complément aux services en ligne, et pour toute demande, les citoyens ont la possibilité de s'adresser à un guichet physique, en présence d'un agent qui pourra les aider à réaliser leurs démarches

Tous les utilisateurs souhaitent que le contact humain soit préservé, soit pour les aider dans leurs démarches administratives, soit en cas de problème technique sur les plateformes numériques.

Cette mesure est également avancée dans la récente étude de Lire et Ecrire[3] : « *Un réseau dense de guichets doit être maintenu pour remédier aux inégalités sociales renforcées autant par la crise sanitaire que par la dématérialisation qu'elle accentue. Les agents de première ligne des services publics disposent d'une expérience et d'un savoir-faire qui ne peut pas être remplacé si facilement, ni par un outil, ni par un aidant numérique à qui il serait impossible d'être au courant de toutes les subtilités des procédures administratives en matière de déclaration d'impôts, de demandes d'allocation ou d'inscription à l'école…* ».

Si ces permanences nécessitent une prise de rendez-vous, celle-ci ne doit pas être prise uniquement en ligne, conformément à la recommandation 35.

Il est conseillé que le personnel présent aux guichets soit sensibilisé et formé à l'accueil des personnes en situation de handicap (et aux différents types de handicaps). Lors de ces permanences aux guichets, une interprétation en Langue des Signes (en présentiel ou à distance) doit pouvoir être proposée aux personnes sourdes afin qu'elles puissent communiquer avec le personnel du service public.

[3] Louise Culot, « Accessibilité numérique pour les personnes illettrées : la loi ne garantit pas l'égalité », Lire et Ecrire communauté française, février 2022.

B. Accessibilité numérique

Recommandation 2 : Le site respecte les normes d'accessibilité numérique et contient une déclaration d'accessibilité conforme, rédigée ou validée par un organisme compétent

Conformément à l'ordonnance du 4 octobre 2018 relative à l'accessibilité des sites internet et des applications mobiles des organismes publics régionaux et des communes, le site internet public doit répondre aux normes d'accessibilité numérique WCAG 2.1 AA.

Certains aspects et critères techniques sont abordés dans ce rapport, mais ceux-ci ne sont pas exhaustifs et une analyse complète des sites, au regard des normes d'accessibilité numérique, est indispensable car elle constitue un socle indispensable à l'accessibilité des sites internet.

Pour atteindre cet objectif, il est conseillé d'être accompagné, depuis le début de la conception, par un organisme compétent.

Par ailleurs, à sa mise en ligne, la déclaration d'accessibilité conforme doit être présente sur le site. Celle-ci doit être rédigée ou validée par un organisme justifiant de compétences en accessibilité numérique.

Pour aller plus loin

Référentiel technique, en deuxième partie de ce rapport.

C. Mise en page

Recommandation 3 : Uniformiser la mise en page de tous les services publics numérisés

Lorsque les citoyens doivent effectuer des actions telles que la recherche, la connexion ou la déconnexion, l'accès à son compte, le choix de la langue… il est plus facile de chercher un lien ou un bouton aux mêmes endroits sur les différents services publics en ligne.

En vue de permettre aux utilisateurs de trouver systématiquement les éléments aux mêmes endroits sur les sites, il est recommandé de respecter une mise en page uniforme sur tous les services publics en ligne. Toutes les procédures de base qui constituent tout SPN (connexion, déconnexion, moteur de recherche, compte personnel, prise de rendez-vous, aide et contact, le sitemap…) doivent être placées systématiquement au même endroit.

Une uniformisation de la structure et de la mise en page des SPN permettra aux utilisateurs de faciliter leur apprentissage à l'utilisation de ces sites et une mémorisation plus facile et rapide des manipulations pour entamer toute démarche administrative.

Pour aller plus loin

Aide technique sur la navigation au sein de la page

WCGA 2.1 AA – Navigation cohérente – critère 3.2.3 (lien externe)

Recommandation 4 : La couleur d'un texte doit être suffisamment contrastée avec la couleur de fond

Le respect des contrastes est très important pour permettre la lecture des éléments pour les personnes malvoyantes, âgées ou toute personne qui n'a pas une source de lumière suffisante.

Cette recommandation s'applique également aux sous-titres des vidéos par exemple.

Il en va de même pour le focus du navigateur : sa visibilité doit être assurée sur tous les composants et ne doit pas être masquée par des ombres et par des images de fond ayant la même couleur que le focus.

Pour aller plus loin

WCAG 2.1 AA – Contraste minimum – critère 1.4.3 - (lien externe)

Application Colour Contrast Analyser – (lien externe)

Recommandation 5 : Utiliser une police accessible et veiller à ce que les tailles de caractères ne soient pas figées

Une taille de police trop petite rajoute une fatigue supplémentaire aux utilisateurs voyants et malvoyants qui doivent effectuer leurs démarches administratives en ligne. Il faut donc privilégier de grandes tailles de police de caractères.

Le corps du texte devrait être présenté dans une taille équivalente à 14 à 16 pixels tout en utilisant des valeurs élastiques (em, %...) pour permettre à l'utilisateur de zoomer ou dézoomer le texte en fonction de ses besoins.

De plus, il faut choisir une police sans empattement qui permet entre autres aux personnes dyslexiques de ne pas confondre certaines lettres.

Astuce : Une bonne pratique consiste à tester la suite de caractères aiIlL1 dans les différentes polices pour s'assurer qu'aucun caractère ne peut être confondu avec un autre.

Pour aller plus loin

Quelle est la meilleure police pour les dyslexiques ? - Site de Culture Dys (lien externe)

A guide to Understanding What Makes a Typeface Accessible – Medium.com (lien externe)

Atkinson Hyperlegible Font – brailleinstitute.org (lien externe)

Testing fonts for accessibility – UX Collective (lien externe)

Typographie Luciole (lien externe)

Digital typography and accessibility (lien externe)

Practical Typography (lien externe)

Si l'utilisateur fait usage d'outils d'agrandissement de police et d'interlignage, le cadre et la police des boutons ne doivent pas être figés et doivent s'adapter en fonction de l'agrandissement ou du rétrécissement.

Pour aller plus loin

WCAG 2.1 AA – Redimensionnement du texte – critère 1.4.4 - (lien externe)

WCAG 2.1 AA – Espacement du texte – critère 1.4.12 - (lien externe)

Recommandation 6 : L'information ne doit pas être donnée uniquement par la couleur ou la forme

Une information ne doit pas être donnée uniquement par la couleur ou la forme sous peine de ne pas être perceptible par les utilisateurs déficients visuels.

De plus, l'indication de la page courante dans les onglets ne peut pas être simplement un changement de couleur de texte, mais doit être combinée avec une mise en forme au choix, cadre, soulignement, …

Pour aller plus loin

WCAG 2.1 AA – Utilisation de la couleur – critère 1.4.1 - (lien externe)

Aide technique sur les labels significatifs et des libellés sous forme d'icônes

Recommandation 7 : Adopter le Responsive Design

Un site qui n'est pas responsive se remarque par son manque d'adaptabilité sur le support (GSM ou tablette) : il faut zoomer ou dézoomer la page du site, car elle n'est pas adaptée à l'écran, certains boutons ne fonctionnent pas, des fenêtres, des menus ou des listes déroulantes ne s'affichent pas correctement ou ne fonctionnent pas…

D. Structure et navigation

Recommandation 8 : Prévoir au moins les 3 chemins d'accès suivants pour naviguer : le menu de navigation, le plan de site et un moteur de recherche

Pour que l'utilisateur accède à l'information qu'il cherche, plusieurs moyens doivent lui être proposés, dont au moins le menu de navigation, un moteur de recherche et un plan du site.

Le plan du site n'est souvent plus implémenté sur les sites, c'est pourtant un système de navigation qui aide grandement l'utilisateur à visualiser la structure du site. Ce plan du site doit être accessible depuis le même endroit sur chaque page du site.

Cependant, la multiplication des chemins pour trouver une même information ou un formulaire, si elle est imaginée pour faciliter la recherche, peut générer une surcharge d'information et rendre l'accès à l'information plus complexe. Il arrive alors que le visiteur soit totalement perdu et démotivé de poursuivre sa démarche.

Il est dès lors important de simplifier et standardiser les procédures (à l'extrême) et, lors de la conception ou de l'implémentation de toute nouvelle procédure, de tester les fonctionnalités avec un échantillon des publics ayant pris part aux évaluations dans le cadre de la présente mission (voir recommandation 37)

Pour aller plus loin

Aide technique sur les aspects techniques de la navigation

RGAA – Navigation – Systèmes de navigation – Critère 12.1 (lien externe)

Recommandation 9 : Prévoir un moteur de recherche performant et utilisable à la voix

Le moteur de recherche est un moyen très prisé par les utilisateurs. Certains l'utilisent parfois même comme seul chemin pour accéder aux informations. Il est donc indispensable qu'il soit efficace et performant.

Pour cela, le moteur de recherche doit accepter les fautes d'orthographe, prendre en compte les synonymes, tenir compte de l'écriture phonétique, proposer des suggestions de mots et, si possible, être utilisable à la voix.

Un filtre doit être ajouté pour pouvoir trier les résultats d'une recherche.

Lorsqu'un utilisateur effectue une recherche sur le site via le moteur de recherche, le focus du clavier doit se positionner sur le premier résultat de recherche et ne doit pas se repositionner en haut de la page.

Par ailleurs, lorsque les délais de chargement des résultats sont plus longs, si le focus n'est pas instantané, l'utilisateur doit être averti que les résultats sont en train d'être chargés.

Pour les utilisateurs qui éprouvent des difficultés avec le français écrit ou tout simplement pour gagner du temps, activer l'autocomplétion peut s'avérer très pratique.

L'autocomplétion est une fonctionnalité qui propose des mots à l'utilisateur à partir des premiers caractères saisis ou à partir des informations personnelles de l'utilisateur déjà préenregistrées sur son navigateur.

Pour aller plus loin

Aide technique sur l'autocomplétion

Les moteurs de recherche doivent par ailleurs respecter les critères d'accessibilité applicables aux formulaires, à la sémantique et à la navigation.

Notamment :

- Les labels accessibles
- Les instructions et autres messages d'information
- Le title de la page
- L'accessibilité au clavier

Recommandation 10 : Respecter les standards de structuration des sites internet

Une bonne structure du site est indispensable pour les personnes utilisatrices de technologies d'assistance et pour celles utilisant uniquement le clavier. Comme elles n'utilisent pas la souris, elles dépendent de cette structure pour pouvoir naviguer de manière cohérente sur le site.

Par respect de la structuration, il est entendu de respecter les niveaux de titres, la manière de concevoir les listes à puce, d'identifier les zones de navigation du site menu, zone principale, pied de page,… tous ces éléments ont un code dédié pour que les utilisateurs puissent au besoin se déplacer de zone en zone, tout en évitant les zones non désirées.

Pour aller plus loin

Aide technique sur la sémantique et la structure

Aide technique sur les listes

Recommandation 11 : Implémenter les liens d'évitement

Pour les utilisateurs aveugles, malvoyants et utilisateurs du clavier seulement, des liens d'évitement doivent être prévus afin que la synthèse vocale ne répète pas systématiquement toutes les rubriques avant d'atteindre le contenu souhaité.

Ce lien d'évitement doit être le premier dans l'ordre de tabulation de chaque page web et doit être présent sur chaque page du site. S'il est caché, il doit au moins être rendu visible à la prise de focus.

Pour aller plus loin

Aide technique sur les liens d'évitements (skiplinks)

Recommandation 12 : Afficher une information permettant de connaître son emplacement dans l'arborescence du site

Afin de permettre aux visiteurs de se repérer dans l'avancement de leurs démarches, il est important d'afficher, sur toutes les pages ainsi que dans les formulaires administratifs à compléter, une information permettant de connaître son emplacement dans l'arborescence du site (un fil d'Ariane par exemple). Cette fonctionnalité facilite la navigation et la représentation mentale de la structure du site et de l'état d'avancement de toute demande administrative.

Pour aller plus loin

Aide technique sur la navigation entre plusieurs pages

Aide technique sur l'identification des étapes dans les formulaires progressifs

Recommandation 13 : L'entièreté du site doit être utilisable au clavier

Il arrive très souvent que les personnes utilisant uniquement le clavier ne puissent accéder à certains boutons ou rubriques.

Afin de pouvoir se repérer sur la page, le focus doit être visible sur tous les éléments interactifs atteignables au clavier. Il est également indispensable que tous ces éléments interactifs soient activables via la touche « Entrée » ou « Espace ».

Par ailleurs, la navigation pour les utilisateurs d'une synthèse vocale ou utilisant uniquement le clavier est perturbée si l'ordre de tabulation n'est pas cohérent, c'est-à-dire que le sens de lecture n'est pas respecté (de haut en bas et de gauche à droite pour le français). Ces derniers doivent alors parcourir tout le site avant d'atteindre la rubrique, le bouton, le lien… voulu.

Toute fenêtre contextuelle doit pouvoir être fermée en utilisant le clavier en atteignant le bouton de fermeture ou via la touche ECHAP/ESC.

Enfin, lors de la navigation de l'utilisateur sur l'un des composants interactifs, la tabulation ne doit pas être piégée dans le composant et ne doit pas rester bloquée dans les commandes d'un slider d'images.

Pour aller plus loin

Aide technique sur la visualisation et l'identification des liens

Aide technique sur l'accessibilité du clavier

Recommandation 14 : Utiliser des termes explicites qui décrivent précisément l'action d'un élément (lien/bouton)

Certains boutons et liens sont peu explicites ou compréhensibles pour des utilisateurs déficients visuels.

En effet, pour les personnes voyantes, la disposition de boutons et d'éléments peut communiquer une information supplémentaire, qui n'est pas perceptible par les personnes déficientes visuelles.

De plus, les personnes utilisatrices de technologies d'assistance peuvent naviguer d'un élément (lien ou bouton) à l'autre. Les termes de ces éléments doivent alors être

compréhensibles et descriptifs. Il faut donc éviter d'utiliser des termes comme « En savoir plus », « Plus d'infos », « A lire »,…

Pour aller plus loin

Aide technique sur les Labels

Aide technique sur le nom accessible des liens

E. Contenus rédactionnels

Recommandation 15 : Limiter la longueur des textes et illustrer le contenu avec des visuels

Trop de contenu textuel démotive l'utilisateur, voire le fait fuir, car il doit fournir davantage d'énergie et de concentration pour rechercher l'information souhaitée, qui est noyée dans le site.

Pour éviter cet écueil, il faut limiter la longueur des textes au maximum et utiliser les visuels qui permettent d'une part d'illustrer le contenu textuel du site, voire même d'apporter des précisions et d'autre part d'aérer le texte, ce qui permet une meilleure compréhension du contenu.

Les visuels à utiliser peuvent être des pictogrammes, des images, des captures d'écran pour expliquer des procédures en ligne, …

Pour aller plus loin

WCAG – Comprendre la règle 3.1 -Rendre le contenu lisible et compréhensible (lien externe)

L'information pour tous – Règles européennes pour une information facile à lire et à comprendre (lien externe – PDF 838 KO)

Recommandation 16 : Traduire le site en français, néerlandais et anglais

Ajouter une traduction en anglais des services en ligne permettrait à une grande partie des personnes de langue étrangère qui ne parlent ni ne comprennent le français ou le néerlandais, à réaliser leurs démarches administratives.

Recommandation 17 : La langue doit rester cohérente pour l'utilisateur tout au long de la navigation

Sur un site multilingue, chaque utilisateur peut choisir la langue de son choix. Mais il arrive que certaines pages d'un même site changent subitement de langue en cours de navigation, sans même que cela soit notifié. Les utilisateurs sont alors perdus, désemparés et ne peuvent poursuivre leur requête, car ils ne comprennent pas la nouvelle langue.

Chaque langue doit donc être homogène pour l'utilisateur. Lorsque l'utilisateur est redirigé vers toute autre page (connexion, procédure de paiement,…), la redirection doit amener sur la langue logique de navigation. Si le site est en français, la langue de toutes les pages doit être la version française.

Pour aller plus loin

Aide technique sur la langue de la page

Recommandation 18 : Traduire les actions administratives en FALC et en Langue des Signes

Le français écrit, et plus particulièrement le langage administratif, n'est pas toujours compréhensible pour le citoyen lambda. Celui-ci risque alors de renseigner des informations incorrectes ou incomplètes durant ses démarches administratives.

Pour y remédier, toutes les actions pour lesquelles il est demandé de renseigner des informations (formulaire à remplir, documents à compléter ou à charger,…) doivent être disponibles en Facile à Lire et à Comprendre (FALC) et en Langue des Signes (LSFB) via des capsules vidéo. Il faut aussi expliquer toutes les étapes de la démarche, son objectif, la manière dont il faut répondre et soumettre la demande,…

La présence des vidéos en Langue des Signes doit être indiquée clairement sur le site. Le pictogramme de la LSFB, représenté par des mains, doit être présent sur toutes les pages du site, toujours au même endroit. (ex : le site d'Unia)

À nouveau, lors de la conception, ou de l'implémentation de toute nouvelle procédure, il y a lieu de tester les contenus avec un échantillon des publics ayant pris part aux évaluations dans le cadre de la présente mission (voir recommandation 37).

Pour aller plus loin

L'information pour tous – Règles européennes pour une information facile à lire et à comprendre (lien externe – PDF 838 KO)

Les règles européennes pour des informations faciles à lire – Inclusion Europe (lien externe)

Référentiel français pour une rédaction de contenus alternatifs en langage simplifié (lien externe)

Facile à Lire et à Comprendre (FALC) – FALC.be : un service pour rendre l'information accessible à tous en français Traduction, formation et promotion (lien externe)

Wablieft vzw : service pour rendre l'information accessible à tous en néerlandais Traduction et formation (lien externe)

LSFB : un service de traduction pour des capsules vidéo en Langue des Signes francophone (lien externe)

Visual box : service de traduction pour des capsules vidéo en Langue des Signes néerlandophone (lien externe)

Recommandation 19 : Ajouter une alternative textuelle pour les images qui contiennent une information

Pour les personnes utilisatrices de synthèse vocale, seules les images qui contiennent une information non présente dans la page doivent être rendues accessibles via une alternative textuelle. À l'inverse, les images décoratives ne doivent pas en avoir.

Pour aller plus loin

Aide technique sur les alternatives des images

F. Création d'un compte, connexion et déconnexion

Recommandation 20 : Mettre en œuvre la procédure de connexion via CSAM

La création d'un compte peut s'avérer fastidieuse en raison notamment de l'obligation de choisir un mot de passe complexe, long et difficile à retenir, de la confirmation de la création du compte par mail, de la procédure de récupération d'un mot de passe oublié,…

Pour éviter de devoir se créer un compte différent pour chaque SPN, il faut uniformiser le moyen de se connecter aux différents services publics via les clés numériques du CSAM.

Cependant, pour que cette méthode soit utilisable par tous, des explications d'utilisation (et de création de compte pour Itsme entre autres) doivent être disponibles en FALC et dans des vidéos en Langue des Signes.

Une adresse mail et un numéro d'appel, avec interprétation à distance en Langue des Signes, doivent être renseignés en cas de problème technique ou pour toute question liée à l'utilisation de cette clé d'identification.

Recommandation 21 : Prévoir un message qui confirme l'état de la connexion/déconnexion

Une fois l'utilisateur connecté/déconnecté, il est important d'afficher clairement la confirmation du succès ou de l'échec de sa connexion/déconnexion. Cette recommandation est applicable à toutes les actions telles que la procédure de paiement, la prise de rendez-vous,…

Pour aller plus loin

Aide technique sur l'identification des feedbacks

Recommandation 22 : Prévoir un message d'alerte si un utilisateur ferme une fenêtre sans s'être déconnecté au préalable

Les personnes peu familières avec les outils informatiques ne possèdent pas certains réflexes, notamment celui de se déconnecter avant de fermer la page d'un site. Ceci peut générer des soucis de sécurité si la personne s'est connectée sur un ordinateur partagé (bibliothèque, EPN…).

Un message d'alerte doit s'afficher lorsque l'utilisateur tente de fermer la fenêtre ou l'onglet, lui indiquant que cette action va le déconnecter immédiatement ou endéans un certain délai. Le cas échéant, l'utilisateur doit être informé s'il s'apprête à fermer la page, mais qu'il restera cependant connecté et lui indiquer alors la procédure de déconnexion.

Pour aller plus loin

Aide technique sur l'identification des feedbacks

Aide technique sur les alertes et les annonces

Aide technique sur les changements de contexte

Recommandation 23 : Allonger la durée maximale pour la procédure de connexion

Les utilisateurs disposent de 3 minutes pour se connecter à un SPN via l'application Itsme. Cette limite de temps est trop courte pour différents publics, surtout s'il y a une vérification supplémentaire via des pictogrammes à faire correspondre entre le téléphone et l'ordinateur. Il faut donc allonger cette durée pour laisser un temps qui soit confortable à l'utilisateur pour se connecter.

Pour aller plus loin

WCAG 2.1 – Utilisable – Règles pour le délai suffisant (lien externe)

G. Encoder des informations et remplir des formulaires

Recommandation 24 : Assurer l'accessibilité des formulaires

Les formulaires comportent régulièrement des difficultés pour les utilisateurs de clavier ou de synthèse vocale. Il peut être compliqué de savoir quelle information est attendue dans les champs, de savoir où on en est dans le formulaire ou encore de trouver le bouton pour envoyer sa réponse.

Les mesures de sécurité de type « authentification par question-réponse » (captcha par exemple) sont également souvent inaccessibles pour une série d'utilisateurs.

Tous les champs de formulaires (champs de saisie, des boutons radio, cases à cocher ou autre) doivent avoir une étiquette visible liée visuellement et techniquement.

Pour aller plus loin

Aide technique sur les formulaires accessibles

Les CAPTCHAs et l'accessibilité – Orange.com (lien externe)

Recommandation 25 : Demander uniquement les informations strictement nécessaires

Plus il y a d'informations à encoder, plus le risque est grand que l'utilisateur se trompe dans l'encodage de ses données ou abandonne sa démarche.

Il faut donc veiller à ne demander que les informations strictement nécessaires dans les formulaires, conformément au Règlement Général de Protection des Données (RGPD).

Recommandation 26 : Permettre d'introduire et de modifier manuellement ses données personnelles

Les utilisateurs ne se retrouvent pas toujours dans les choix proposés dans les listes déroulantes des formulaires ou champs à compléter. Ils risquent alors d'encoder une information approximative, voire incorrecte.

Pour cette raison, il faut permettre aux utilisateurs d'introduire manuellement un maximum d'informations (via un champ « autre » par exemple).

De plus, il ne faut pas rendre systématiquement toutes les informations obligatoires, car il peut arriver que l'utilisateur n'ait pas d'information à renseigner. La mention « Obligatoire » est par ailleurs privilégiée à l'astérisque (*) qui n'est pas compris par tout le monde.

L'utilisateur doit également pouvoir modifier un maximum de ses données personnelles : si certaines données ne peuvent être modifiées instantanément directement sur le site, il faut permettre à l'utilisateur d'introduire facilement (via un bouton par exemple) une demande de modification de ses données.

Pour aller plus loin

Aide technique sur les labels explicites

Aide technique sur les instructions des champs de formulaires

Aide technique sur les champs requis

Aide technique sur les formulaires dynamiques

Aide sur les design patterns WAI-ARIA

Recommandation 27 : Préciser le format des informations demandées

Le format des informations demandées (téléphone, adresse, date de naissance,…) varie d'un site à l'autre. Des utilisateurs perdent parfois beaucoup de temps à introduire leurs données, car ils ne connaissent pas le format attendu.

Afin que les utilisateurs puissent encoder correctement les informations demandées, il faut préciser le format attendu avec des exemples à l'appui.

Pour un format de date, de numéro de téléphone, de format de mail, il est demandé de donner des exemples concrets avec des exemples factices : pour une adresse mail, donner l'exemple de jean.dupont@gmail.com ou pour un numéro de gsm +32 490 11 12 13.

Attention : il faut éviter les formats du style 04XX/ XX XX XX ou email@example.com qui ne sont pas toujours compréhensibles.

Pour aller plus loin

Aide technique sur les labels explicites

Aide technique sur les instructions des champs de formulaires

Aide technique sur la prévention des erreurs

Recommandation 28 : En cas d'erreur, indiquer clairement l'information à modifier

Si l'utilisateur a oublié ou mal encodé une information, un message d'erreur apparait pour le lui signaler. Cependant, ce message d'erreur n'est pas toujours clair ou visible.

Les messages d'erreurs doivent être correctement reliés aux champs de formulaire avec un message explicite : « Il est obligatoire d'écrire son prénom » au lieu de « Ce champ est obligatoire ». Il doit être également fait mention, avant remplissage des champs, de la nature obligatoire.

Par ailleurs, le focus du clavier doit se positionner sur le premier champ de formulaire qui est erroné pour pouvoir être repéré directement.

Pour aller plus loin

Aide technique sur les labels explicites

Aide technique sur les instructions des champs de formulaires

Aide technique sur la prévention et la correction des erreurs

Recommandation 29 : Prévoir un message qui confirme l'état de soumission du formulaire

Quand un utilisateur remplit un formulaire, il n'est pas toujours bien indiqué si ce dernier a été effectivement soumis. Il arrive donc que l'utilisateur réintroduise son formulaire, croyant que sa première tentative a échoué.

Il faut donc qu'un message indique clairement l'état de soumission du formulaire, et ce, en haut de page.

Pour aller plus loin

Aide technique sur la confirmation d'envoi de formulaires

Aide technique sur le title de la page

Aide technique sur les alertes et annonces sur la page

H. Charger et télécharger des documents

Recommandation 30 : Identifier la nature et le poids du document téléchargeable

Pour éviter qu'un document ne surcharge l'outil informatique de l'utilisateur, celui-ci doit connaître au préalable la nature et le poids du document afin d'évaluer s'il peut télécharger le document en question.

Il faut donc identifier la nature (PDF, Word, Powerpoint,…) et le poids du document (Ko, Mo, Go,…). Si le document est dans une autre langue que celle de la page, la langue du document doit également être précisée.

Pour aller plus loin

Aide technique sur les lies externes et de téléchargement

Recommandation 31 : Expliquer la numérisation et le téléversement des documents

Les personnes qui ne sont pas familières avec les outils informatiques ne savent ni comment numériser ni comment téléverser un document.

Des explications doivent donc être données en FALC et illustrées par des captures d'écran pour expliquer ces deux opérations.

I. Effectuer des démarches payantes

Recommandation 32 : Proposer le paiement par carte de débit (Bancontact) et de crédit (Visa et Mastercard)

Toute démarche payante doit être annoncée à l'utilisateur avant qu'il ne se lance dans l'introduction de sa demande administrative. En effet, certaines personnes ne sont pas outillées pour réaliser des paiements en ligne et ne pourraient dès lors pas finaliser leur démarche.

Pour toute démarche administrative payante, il faut proposer au minimum le paiement par Bancontact, Visa et Mastercard.

Pour aller plus loin

Règle Opquast 47 – Les moyens de paiement acceptés et les procédures correspondantes sont indiqués (lien externe)

Règle Opquast 57 – Le site accepte au moins deux moyens de paiement (lien externe)

Aide technique sur les alternatives d'images

Recommandation 33 : Prévoir un message qui confirme l'état du paiement en ligne

De la même manière que pour la connexion (recommandation 21), un message doit indiquer que le paiement a été effectué avec succès ou non, afin d'éviter que l'utilisateur ne répète l'opération inutilement.

Par ailleurs, différents publics se méfient du niveau de sécurité des sites internet et plus particulièrement des paiements en ligne. Comme indiqué dans la recommandation 18, il faut donner des explications sur la procédure de paiement et y préciser le niveau de sécurité du site, tant en FALC que dans une vidéo traduite en Langue des Signes sous-titrée.

Pour aller plus loin

Règle Opquast 60 – La référence de la transaction est affichée au client après la validation de sa commande (lien externe)

Règle Opquast 62 – Chaque facturation fait l'objet d'un mail de confirmation (lien externe)

Aide technique sur l'identification des feedbacks

Aide technique sur la confirmation d'envoi de formulaires

Aide technique sur le title de la page

Aide technique sur les alertes et annonces sur la page

J. Multimédia (vidéos et fichiers sonores)

Recommandation 34 : Les vidéos et fichiers sonores doivent systématiquement être accessibles à tous

Pour tout contenu multimédia ayant des informations sonores, un sous-titrage synchronisé est attendu dans la langue de la vidéo.

Les sous-titres ne devront pas être incrustés dans la vidéo, mais ajoutés séparément via un fichier de sous-titrage (.srt ou autre) pour en permettre la personnalisation (adaptation de la taille par exemple) par les utilisateurs qui en auraient besoin.

Par ailleurs, il faut prévoir un lecteur vidéo accessible permettant d'afficher ou masquer les sous-titres.

Dans les sous-titres, toutes les infos utiles devront s'y retrouver, ainsi que des informations situationnelles ou contextuelles telles que les éléments sonores autres que la voix utile à la compréhension de la vidéo.

Le sous-titrage est complémentaire à la transcription textuelle.

Une audiodescription doit également être apportée si des éléments visuels ne sont pas décrits sur le site ou dans la vidéo alors qu'ils apportent une information.

Tous les vidéos et fichiers sonores doivent être traduits en Langue des Signes.

Si une vidéo n'illustre pas un contenu textuel, il faut prévoir une transcription textuelle pour les utilisateurs qui ont du mal à suivre le sous-titrage d'une vidéo. Elle doit être disponible aux abords de la vidéo, soit directement sous la vidéo, soit via un lien pointant vers elle. Toute information sonore utile devra y être renseignée.

Pour aller plus loin

WCAG 2.1 – Perceptible – Accessibilité des Média temporels (lien externe)

Notice 8.3 AcceDe Web – Savoir gérer les vidéos accessibles (lien externe)

W3C-WAI – Videos caption (lien externe)

W3C-WAI – Making Audio and Video Media Accessible (lien externe)

Comment rendre les supports multimédias accessibles à tous ? – Emmanuelle Aboaf (lien externe)

K. Prise de rendez-vous

Recommandation 35 : La prise de rendez-vous doit pouvoir s'effectuer par téléphone et via un calendrier en ligne

Comme déjà expliqué précédemment, uniformiser les procédures de base des SPN permet à l'utilisateur d'apprendre à les reproduire plus rapidement et plus facilement. Cela vaut donc aussi pour la prise de rendez-vous.

Cette procédure doit pouvoir s'effectuer par téléphone et via un agenda en ligne pour permettre à l'utilisateur de choisir lui-même le jour et la date de rendez-vous. Cette fonctionnalité permet aux personnes sourdes de choisir une date et une heure de rendez-vous en fonction des disponibilités d'un interprète (pour une interprétation à distance) ou d'un proche.

Conformément aux recommandations 21, 29 et 33, un message doit signaler à l'utilisateur la confirmation du succès ou de l'échec de la prise de rendez-vous. Un mail de confirmation doit également être envoyé afin que le citoyen ait une trace écrite de son rendez-vous.

Pour aller plus loin

Aide technique sur l'identification des feedbacks

Aide technique sur la confirmation d'envoi de formulaires

Aide technique sur le title de la page

Aide technique sur les alertes et annonces sur la page

Aide technique sur le respect des Design patterns WAI-ARIA

L. Contact

Recommandation 36 : Proposer au minimum le mail ou formulaire de contact accessible et le téléphone (complété d'un dispositif d'interprétation à distance en Langue des Signes)

Comme pour toute démarche administrative effectuée en présentiel, il est possible que le citoyen rencontre des embuches durant sa navigation en ligne : un problème technique est vite arrivé, un formulaire plus complexe peut vite devenir incompréhensible,…

Afin d'assurer une assistance à tous les citoyens, il faut proposer au minimum ces deux moyens de contact : le mail et/ou le formulaire de contact et le téléphone avec interprétation à distance en Langue des Signes.

Pour aller plus loin

Règle Opquast 100 – L'adresse complète et le numéro de téléphone sont disponibles depuis toutes les pages du site (lien externe)

Règle Opquast 102 – Le site propose au moins deux moyens de contact (lien externe)

M. Processus de conception des SPN

Recommandation 37 : Faire tester le SPN par un panel diversifié de citoyens dès la conception

Afin que le SPN soit véritablement accessible pour tous les citoyens, il faut le faire tester par un maximum de groupes de citoyens différents, dont ceux exposés en page 3, dès la phase de conception.

Comme le souligne la vidéo « *Service design : what is it ?* [4]» disponible dans les outils du Playbook du BOSA, il faut mettre le citoyen au premier plan dès la conception de tout projet : plus vite un projet est confié à un utilisateur, plus vite il sera possible d'apporter des ajustements si un problème se présente. Une nouvelle phase de test peut par ailleurs être organisée avant la mise en ligne officielle du SPN afin de s'assurer que les ajustements ont bien été effectués et répondent véritablement aux besoins de tous les utilisateurs.

Faire tester le site par ces différents publics permet de renforcer l'efficacité du service, d'assurer son accessibilité et enfin, de faire remonter les écueils tant fonctionnels que techniques[5].

Recommandation 38 : Réaliser un mode d'emploi des SPN

Une fois que la structure et la mise en page de tous les SPN seront uniformes, un mode d'emploi unique et universel devra être réalisé pour expliquer comment naviguer sur les sites et comment réaliser ses démarches administratives en ligne.

Ce tutoriel doit être proposé en format vidéo (traduite en FALC et en Langue des Signes et sous-titrée) et en format papier, téléchargeable et imprimable, rédigé en FALC. Les deux formats doivent être agrémentés d'illustrations (captures d'écran, pictogrammes,...).

Recommandation 39 : Former les professionnels aux recommandations inclusives

Afin que les recommandations inclusives soient correctement implémentées sur les SPN, tous les professionnels en charge de ces sites (webdesigners, graphistes, rédacteurs de contenus,…) doivent être formés à l'accessibilité numérique.

Ces formations doivent être organisées à chaque engagement ou renouvellement de poste, dans le but d'assurer cette continuité dans la mise en accessibilité des SPN.

[4] Vidéo disponible sur le site du Playbook du BOSA. En ligne : https://digitalopen.belgium.be/fr/playbook/tools/conception-de-services
[5] Faure, L & Brotcorne, P (2021). *Guide pour une conception inclusive des services numériques.* Idéalic.be. p.18.

N. Une plateforme test pour chaque SPN

Recommandation 40 : Mettre à disposition des organes d'accompagnement et de formation une plateforme test en ligne permettant aux utilisateurs d'essayer les services en ligne lors de formations

Afin de permettre l'apprentissage de l'utilisation des SPN, les services publics doivent mettre à disposition une exacte réplique (à jour) de leur site internet en version « test ».

Ces sites dédiés à l'apprentissage permettront aux services d'accompagnement et de formation de réaliser avec les utilisateurs des tests sur les plateformes, sans apporter de changement sur les dossiers personnels des citoyens, sans impliquer de payement réel, ou encore de surcharge administrative inutile pour les services lors de demandes factices.

Pour cela, la Région devrait mettre en place une procédure simple de consultation du panel sur demande des concepteurs des sites internet des services publics, via un marché-cadre ou une centrale d'achat.

De plus, la vérification de l'application des recommandations inclusives par des organismes compétents permettra de valider le caractère inclusif de ces services.

III. Partie II : Référentiel technique pour les développeurs

O. Les formulaires accessibles

Labels (étiquettes)

Chaque champ de formulaire doit avoir un champ compréhensible par tous les utilisateurs, qu'ils puissent ou non consulter la page avec leurs yeux.

a) Labels sémantiques

Les labels doivent être techniquement associés à leurs éléments

Chaque champ de formulaire ou contrôle doit avoir un label aussi appelé « nom accessible ». Ce label doit être techniquement associé à son champ sans ambiguïté quant à son sens.

Il existe plusieurs possibilités pour créer des labels accessibles. La meilleure manière consiste à utiliser les balises HTML imaginées à cet effet. C'est la solution la plus robuste, reconnue par tous les navigateurs et les technologies d'assistance. Par ailleurs, leur utilisation permet à l'utilisateur de positionner le focus dans le champ en cliquant sur le label. La surface de clic est ainsi élargie et facilite la navigation des personnes qui rencontrent des difficultés motrices ou visuelles qui les empêchent de viser une cible trop petite.

Il existe une hiérarchie entre les méthodes pour fournir un nom accessible à un champ :

- Aria-labelledby : Les technologies d'assistance utilisent le texte référencé par l'ID de l'attribut aria-labelledby comme nom accessible. Il faut noter que même si le texte du aria-labelledby est visible à l'écran, un clic sur ce texte ne positionne pas le focus sur le champ correspondant. A utiliser donc uniquement si le <label> ne fonctionne pas.
- Aria-label : le texte de l'attribut aria-label est utilisé comme nom accessible s'il n'existe pas de aria-labelledby. A noter que le texte du aria-label est complètement invisible et n'est donc utile que pour les utilisateurs des lecteurs d'écran. Il n'est donc pas approprié dans la plupart des circonstances.
- <label> : Cette méthode est la plus recommandée dans toutes les circonstances. Le contenu du label sera utilisé comme nom accessible en cas d'absence d'aria-label ou de aria-labelledby.
- Attribut title : Cet attribut ne peut contenir d'informations essentielles et ne peut donc être utilisé comme méthode principale pour donner un nom accessible. Le contenu du title n'est rendu visible qu'au passage de la souris et pas à la prise de focus. Les utilisateurs qui n'ont pas accès à la souris (la plupart des technologies d'assistance et utilisateurs de clavier exclusif) n'ont pas accès à l'information.
- Placeholder : Le placeholder peut techniquement fournir un nom accessible mais cette technique n'est pas recommandée pour les raisons suivantes. Le contenu disparait dès que l'utilisateur commence à écrire dans le champ ; Le style par défaut ne rencontre pas les critères de contraste minimum ; Le placeholder ne peut contenir que du contenu non essentiel.

Bons exemples :

Labels explicites

Méthode recommandée en toutes circonstances. Les champs sont associés explicitement avec leur label respectif.

<p>

```
  <label for="fname_a">First Name:</label>
  <input type="text" name="fname_a" id="fname_a">
</p>
<p>
  <label for="lname_a">Last Name:</label>
  <input type="text" name="lname_a" id="lname_a">
</p>
```

Labels implicites

L'association implicite est créée par la mise en place du champ à l'intérieur du label.

Cette méthode peut ne pas être entièrement supportée par toutes les combinaisons de navigateurs / lecteurs d'écran. Quelques cas ont été remontés pour Firefox + NVDA.

```
<p>
  <label>First Name: <input type="text" name="fname1"></label>
</p>
<p>
  <label>Last Name: <input type="text" name="lname1"></label>
</p>
```

Aria-labelledby comme input text

Le champ texte contient le nom accessible en liant la référence de son id à l'attribut aria-labelledby présent dans le champ input.

Cette technique est conforme mais présente un inconvénient : le clic dans le champ texte ne positionne pas le focus dans le champ input à l'inverse de la méthode du champ explicite. Dès lors, cette méthode ne peut être utilisée que si le <label> ne fonctionne pas pour une raison ou une autre.

```
<p>
<span id="Nickname">Nickname:</span> <input type="text" aria-labelledby="Nickname">
</p>
```

Aria-label comme input text

Cet exemple montre un champ de recherche qui utilise l'attribut aria-label pour fournir un label pour les lecteurs d'écran lorsqu'il n'existe pas de label visible. Le champ de recherche est visiblement labellisé grâce au bouton de recherche tout proche. Il s'agit d'un rare cas où il est permis de ne pas implémenter un label visible lié au champ input.

```
<p><input type="text" aria-label="search"> <input type="submit" value="Search"></p>
```

Attribut title comme input text

Dans cet exemple, l'attribut title est utilisé comme label accessible sur le champ texte. Le title est invisible pour les personnes voyantes qui ne pourront en voir le contenu qu'au survol de la souris. Les utilisateurs de clavier exclusif n'auront pas accès à l'information toutefois visible dans le bouton adjacent.

<input type="text" title="search"> <input type="submit" value="Search">

Les labels doivent être présents sous forme de texte

Un label seul n'est pas suffisant s'il ne contient pas de texte clair et pertinent, lisible par une technologie d'assistance. Un label vide n'est pas meilleur que le manque de label. Pire, un tel label peut représenter une source d'erreur de par l'interprétation que pourra en faire la technologie d'assistance ou l'utilisateur. Et si le label ne contient pas de texte, aucune information ne sera transmise à l'utilisateur.

Mauvais exemple : Icones comme labels sans texte alternatif

Des fonts icons sont utilisées pour montrer le format de document que l'utilisateur peut sélectionner dans des cases à cocher. Sans alternative textuelle, la synthèse vocale ne peut pas comprendre le type de format.

<fieldset>

<legend>Choose a file format</legend>

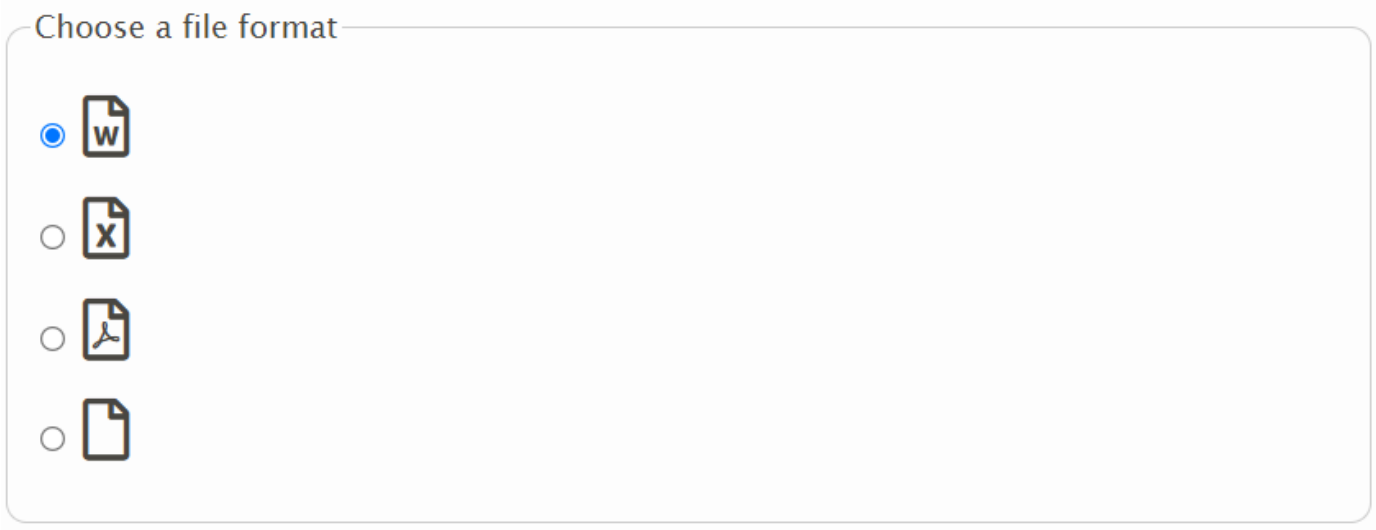
<input type="radio" id="word" name="format9966"> <label for="word"></label>

<input type="radio" id="excel" name="format9966"> <label for="excel"></label>

<input type="radio" id="pdf" name="format9966"> <label for="pdf"></label>

<input type="radio" id="plain" name="format9966"> <label for="plain"></label>

</fieldset>



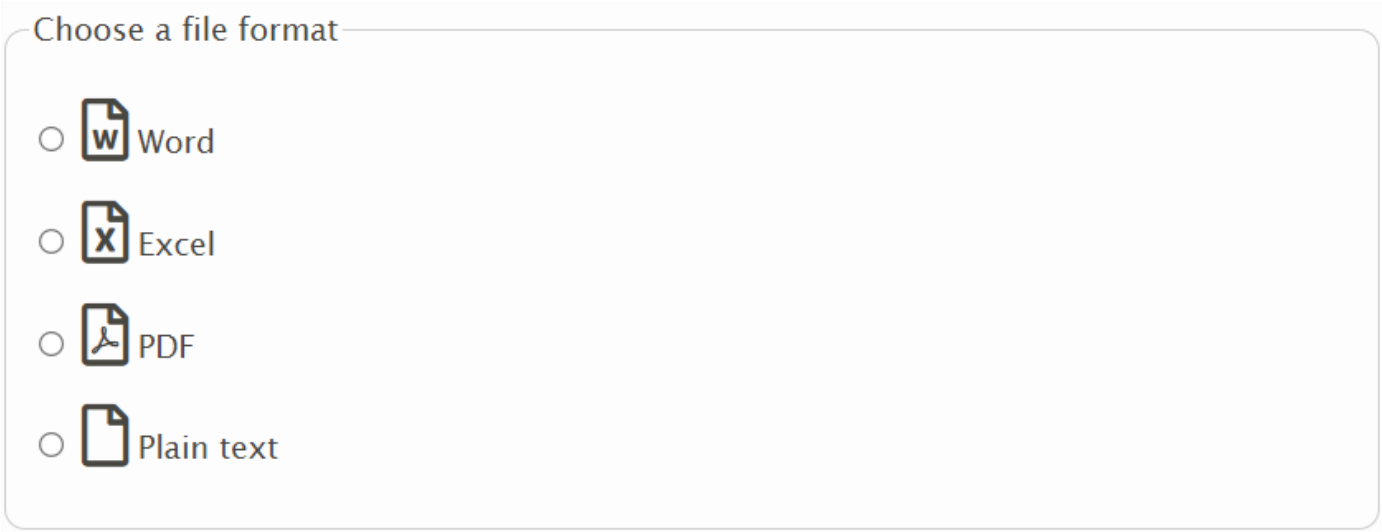
Bon exemple : Icones + labels avec texte

L'usage d'un label contenant du texte permet à la synthèse vocale de comprendre le contenu de chaque case à cocher.

<fieldset>

<legend>Choose a file format</legend>

```
<p><input type="radio" id="word1" name="format9965"> <label for="word1"><span class="far fa-file-word fa-2x"></span> Word</label></p>

<p><input type="radio" id="excel1" name="format9965"> <label for="excel1"><span class="far fa-file-excel fa-2x"></span> Excel</label></p>

<p><input type="radio" id="pdf1" name="format9965"> <label for="pdf1"><span class="far fa-file-pdf fa-2x"></span> PDF</label></p>

<p><input type="radio" id="plain1" name="format9965"> <label for="plain1"><span class="far fa-file fa-2x"></span> Plain text</label></p>

</fieldset>
```



Mauvais exemple : Une image sans alternative textuelle utilisée comme label

Dans cet exemple, le champ semble contenir un label textuel de par l'image qui indique « search ». Une balise label est par ailleurs présente mais elle ne contient pas de texte lisible car l'image ne contient pas d'alternative textuelle.


```
<p>
  <label for="fa-search-no-alt">
    <img src="label-search.png" width="74" height="16">
  </label>
  <input type="text" id="fa-search-no-alt">
</p>
```

Bon exemple : Une image avec une alternative textuelle dans le label

```
<p>
  <label for="fa-search-alt">
    <img src="label-search.png" alt="Search" width="74" height="16">
  </label>
```

```
<input type="text" id="fa-search-alt">
</p>
```

Attention, même si cet exemple est bon d'un point de vue d'un label qui contient une alternative textuelle valable, une image texte est à proscrire en accessibilité. En effet, l'image zoomée perd en qualité et donc en lisibilité et les utilisateurs ne peuvent pas modifier la police, la taille ni la couleur du texte pour répondre à leurs besoins spécifiques.

b) Les labels significatifs

Le texte des labels doit être significatif

Pour pouvoir compléter un formulaire, les utilisateurs ont besoin de labels clairs, informatifs, précis et significatifs. L'internaute doit savoir exactement l'information demandée pour compléter chaque champ sans erreur.

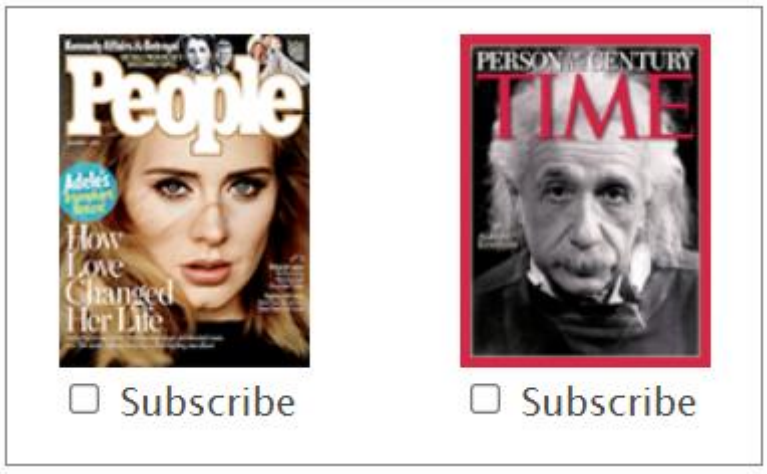
Bon exemple : Un label descriptif

Ce label définit exactement l'information attendue dans le champ. En complément du mot « Nom », la mention du prénom et du nom est renseignée.


```
<label for="name">Name (First and Last):</label> <input type="text" name="name" id="name">
```

Mauvais exemple : Un label ambigu

Dans ce mauvais exemple, les 2 cases à cocher mentionnent seulement le texte « Subscribe ». Les deux choix sont précédés d'une image qui contient la couverture d'un magazine. Ces deux images ne contiennent pas d'alternative textuelle et ne sont pas associées à leur case à cocher respective. De ce fait, l'utilisateur de synthèse vocale n'entendra que le texte « Subscribe » deux fois sans savoir de quel magazine il s'agit.




```
<p style="float:left;margin:0;">
  <img src="assets/images/module-forms/magazine-people.png" width="100" height="133"><br>
  <input type="checkbox" id="people9999"> <label for="people9999">Subscribe</label>
</p>
```

```
<p style="float:right;margin:0;">
  <img src="assets/images/module-forms/magazine-time.png" width="100" height="133"><br>
  <input type="checkbox" id="time9999"> <label for="time9999">Subscribe</label>
</p>
```

Les labels ne doivent pas se baser uniquement sur des références sensorielles pour communiquer l'information.

Se baser uniquement sur des références sensorielles pour communiquer de l'information dans les labels peut empêcher des personnes de comprendre ce qui est demandé. Par exemple, partir du principe que tous les utilisateurs peuvent voir les formes, couleurs ou icônes peut exclure les personnes aveugles, daltoniennes ou avec une basse vision. Voici quelques points à garder à l'esprit :

- Si une couleur est utilisée pour communiquer une information dans un label, s'assurer qu'un autre moyen soit utilisé en plus comme par exemple du texte ou un symbole non ambigu.
- Le contraste de couleur du texte ou de l'icône du label doit respecter le ratio d'accessibilité minimum.
- Toutes les informations communiquées visuellement dans un label doivent pouvoir être comprises par les technologies d'assistance par une technique appropriée.

Mauvais exemple : L'aria-label n'est pas informatif

Dans cet exemple, un champ de texte et son libellé sont suivis par un bouton qui contient la lettre « i ». Visuellement, ce bouton est destiné à donner de l'information sur le champ texte. Mais le aria-label ne contient que l'information « button » qui n'est pas suffisamment claire pour l'utilisateur d'une synthèse vocale.


```
<label for="hue">Favorite hue:</label> <input type="text" id="hue">
<button aria-label="button" id="hueButtonBad">
    <span class="fa fa-info-circle"></span>
</button>
```

Bon exemple : L'aria-label est informatif

L'aria-label contient l'information informative claire.

```
<label for="hue2">Favorite hue:</label> <input type="text" id="hue2">
<button aria-label="What does 'hue' mean?" id="hueButtonGood">
    <span class="fa fa-info-circle"></span>
</button>
```

Mauvais exemple 2 : L'aria-label n'est pas informatif

Même logique que plus haut pour une icône qui contient un point d'interrogation dont le but devrait être d'informer sur le contenu de la case à cocher. L'aria-label contient la phrase « Question mark ».


```
<input type="checkbox" name="dirigible" id="dirigible1">
<label for="dirigible1">I have never flown in a dirigible</label>
<button aria-label="question mark" id="questionButtonBad">
    <span class="fa fa-question-circle"></span>
</button>
```

Bon exemple 2 : L'aria-label est informatif

```
<input type="checkbox" name="dirigible" id="dirigible2">
<label for="dirigible2">I have never flown in a dirigible</label>
<button aria-label="What is a dirigible?" id="questionButtonGood">
    <span class="fa fa-question-circle"></span>
</button>
```

c) *Des libellés sous forme d'icônes*

Les icônes peuvent être utilisés comme libellés sans texte si leur signification est évidente d'elle-même ET si un texte est associé techniquement à l'étiquette pour communiquer l'information aux technologies d'assistance.

Bon exemple : La signification de l'icône est évidente et un texte est associé

Un champ de recherche est suivi d'un bouton qui représente une loupe. Il n'y a pas de texte adjacent visuel qui contient le mot « recherche » car la signification de la loupe est suffisamment claire et régulièrement utilisée sur les sites pour cet usage.

Un aria-label contient le mot "Search" pour informer les utilisateurs de synthèse vocale.

```
<p class="center">
<input type="text" aria-label="Search">
<button id="search-button" aria-label="Search">
    <span class="fa fa-search"></span>
</button>
</p>
```

Mauvais exemple : La signification de l'icône n'est pas évidente même si un texte clair est associé pour les synthèses vocales.

Une icône représente une flasque chimique sans aucun texte visuel à côté. Le contenu du aria-label demande de montrer la formule chimique de de la vitamine C mais ce texte n'est pas visible par les non-utilisateurs de synthèse vocale.

```
<button id="flask-button" aria-label="Show chemical formula for Vitamin C">
    <span class="fa fa-flask fa-2x"></span>
</button>
```

d) Des libellés sous la forme de placeholder

Les placeholder ne doivent pas être utilisés comme seule méthode pour fournir un libellé à un champ de texte.

Beaucoup de designers remplacent le libellé par un placeholder pour gagner de la place sur la page. Le placeholder disparaît au focus dans le champ lorsque l'utilisateur commence à entrer du texte. L'information n'est alors plus visible pour l'utilisateur.

Par ailleurs, le texte du placeholder manque souvent de contraste et certains utilisateurs ne le voient pas. Les designers pourraient alors être tentés de renforcer le contraste mais l'internaute pourrait alors croire qu'il a déjà renseigné une information dans le champ.

Si le placeholder contient des instructions importantes sur la manière dont le champ doit être complété, l'utilisateur perdra l'information une fois les données encodées. Il devra par ailleurs effacer ses propres données pour revoir les instructions en cas de doute.

Les synthèses vocales supportent en général le placeholder mais toutes ne restituent pas l'information lorsqu'une donnée a déjà été encodée.

Dès lors, s'il est utilisé, le placeholder ne peut pas être la seule technique pour étiqueter un champ.

Si le placeholder est utilisé en plus d'une étiquette, la synthèse vocale lit deux fois les informations renseignées : celles contenues dans l'étiquette et celles du placeholder.

Pour toutes ces raisons, nous déconseillons l'utilisation du placeholder même si nous ne pouvons pas invalider l'accessibilité d'un formulaire dans le cadre d'un audit AA si cette technique est combinée à une autre technique d'étiquetage.

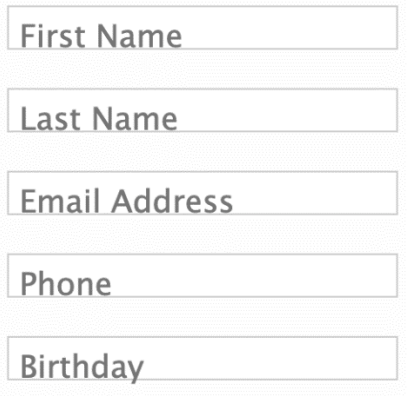
Mauvais exemple : Un placeholder utilisé comme seul libellé

Comme expliqué plus haut, le texte disparaît en entrant des données dans le champ et le contraste par défaut n'est pas suffisant.


```
<p>
<input type="text" placeholder="Last Name"> ,
<input type="text" placeholder="First Name">
<input type="text" placeholder="Middle Name">
</p>
```

Bon exemple : Un placeholder Canada dry conçu avec un vrai label qui se positionne au-dessus du champ au focus et au hover.

Tant que le champ ne contient aucune donnée, le label est positionné dans le champ input grâce au CSS. Au focus et au hover et lorsqu'une information est encodée, le label se positionne au-dessus du champ.



e) Visibilité du libellé

Le libellé doit être visible

Les libellés visibles bénéficient à tous. Des libellés visibles et explicites évitent les erreurs et les pertes de temps inutiles. Les internautes ne devraient pas devoir tenter de deviner ce qu'on attend d'eux.

Bon exemple : Un formulaire visuel qui contient des instructions pour les synthèses vocales

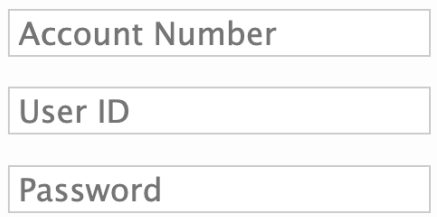
Ce formulaire contient un libellé explicite et montre visuellement, de par sa forme, le nombre de caractères qui peuvent être entrés. Un système permet de ne pas entrer plus de caractères qu'attendus.

Les attributs aria-label communiquent des instructions claires à la synthèse vocale.


```
<p>
<label for="ssn1">Social Security Number:</label>
<input type="text" size="3" maxlength="3" id="ssn1" aria-label="Social Security Number first three digits"> -
<input type="text" size="2" maxlength="2" aria-label="middle two digits"> -
<input type="text" size="4" maxlength="4" aria-label="last four digits">
</p>
```

Mauvais exemple : le placeholder est utilisé en combinaison avec aria-label sans aucun autre libellé.

Nous vous renvoyons au point relatif aux placeholder. Seuls les utilisateurs de synthèse vocale entendront toujours l'information contenue dans aria-label. Cette information sera par ailleurs redondante avec le placeholder.




```
<p><input type="text" placeholder="Account Number" aria-label="Account Number"></p>

<p><input type="text" placeholder="User ID" aria-label="User ID"></p>

<p><input type="password" placeholder="Password" aria-label="Password"></p>
```

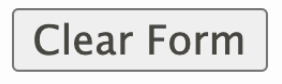
Les composants d'interface dont l'étiquette contient du texte et des textes-images doivent avoir un nom accessible (name) qui contient le texte présenté visuellement.
Si le nom accessible du composant n'est pas le même que celui vu par l'utilisateur, celui-ci ne pourra pas utiliser d'outil vocal pour activer les différents contrôles du formulaire.

Les technologies d'assistance construisent le nom accessible sur base des différents éléments contenus par le composant. Notamment le texte visible, les libellés et les attributs aria-label et aria-labelledby. Chaque élément peut par ailleurs être prioritaire sur les autres. Il faut donc toujours garder les aspects suivants à l'esprit :

1. Vous devriez éviter l'utilisation de texte en image mais en cas d'usage dans un bouton, prévoyez une alternative textuelle identique à l'image texte.
2. Aria-label et aria-labelledby écrasent les autres informations du composant lors de la création du nom accessible. Cela signifie que l'ajout d'un aria-label / aria-labelledby ne renseigne pas une information supplémentaire mais remplace l'information existante d'où l'importance d'y renseigner exactement ce qui est vu par l'utilisateur.

Mauvais exemple : Une étiquette visible dont le texte est différent du nom accessible.

Le libellé visible montre « Clear Form » alors que le nom accessible (name) renseigne « reset ». L'utilisateur qui souhaite utiliser une reconnaissance vocale énoncera « Clear Form » mais le composant ne fonctionnera pas.




```
<p><input type="reset" name="reset" value="Clear Form"></p>
```

Bon exemple : Le texte de l'étiquette est identique au nom accessible


```
<p><input type="submit" name="submit" value="Submit"></p>
```

f) Proximité du libellé avec son champ ou contrôle

Le label devrait être visuellement adjacent à son élément correspondant
Si le label est positionné trop loin de son élément, les utilisateurs de zoom risquent de ne pas comprendre le lien entre les 2.

Dans le cas d'un champ-texte, le label peut être positionné à la gauche du champ mais la position juste au-dessus du champ est encore meilleure.

Bon exemple :


```
<p>
    <label for="fname_a">First Name:</label>
    <input type="text" name="fname_a" id="fname_a">
</p>
<p>
    <label for="lname_a">Last Name:</label>
    <input type="text" name="lname_a" id="lname_a">
</p>
```

Mauvais exemple :


```
<div>
  <div class="labelleft">Username:</div>
  <div class="center"><input type="text" name="usrname" id="usrname"></div>
</div>
```

Le label devrait être adjacent à son élément correspondant dans le DOM

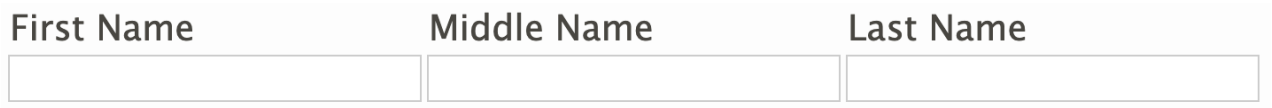
Le lecteur d'écran lit la page dans l'ordre d'apparition des éléments dans le DOM. Si le label est positionné trop loin de son champ, l'utilisateur risque de ne pas faire les liens entre les deux.

Bon exemple :

Voir l'exemple relatif à la proximité visuelle.

Mauvais exemple :

Les éléments du formulaire sont dans un tableau. Puisque le label et son input sont correctement liés grâce au for + id, la consultation via la tabulation permettra à l'utilisateur de faire le lien entre les deux. Cependant, la consultation à l'aide des flèches de tabulation fera entendre à son utilisateur : « first name, middle name, last name, edit, edit, edit ».




```
<table role="presentation">
  <tr>
```

```
    <td><label for="firstname">First Name</label></td>
    <td><label for="middlename">Middle Name</label></td>
    <td><label for="lastname">Last Name</label></td>
  </tr>
  <tr>
    <td><input type="text" id="firstname" name="firstn" size="20"></td>
    <td><input type="text" id="middlename" name="middlen" size="20"></td>
    <td><input type="text" id="lastname" name="lastn" size="20"></td>
  </tr>
</table>
```

g) Plusieurs labels pour un champ

Lorsque plusieurs labels sont utilisés pour un élément, chaque label doit être techniquement associé à son élément correspondant.

Il est parfois nécessaire d'associer plus d'un label dans un champ de formulaire. Le problème est que le label et son champ ont une relation one-to-one dans laquelle un label ne peut être lié qu'à un champ et un champ à un seul label.

Dans ce cas, l'attribut aria-labelledby peut être utilisé pour apporter une solution accessible.

Un exemple classique est l'utilisation de labels multiples dans un formulaire structuré dans un tableau même si, comme mentionné plus haut, cette structure html n'est pas idéale.

Bon exemple de lien de labels multiples pour un seul élément

Dans ce tableau, chaque élément de formulaire doit être lié à plus d'un entête du tableau et chaque entête du tableau doit fournir un label à plus d'un champ de formulaire.

Par exemple, la première case à cocher (checkbox) correspond à « Now » et « Food stamps ». Les utilisateurs de lecteurs d'écran doivent entendre les deux labels lorsqu'ils tabulent dans la cellule « Now / Food stamps ». L'entête « Now » s'applique également à tous les éléments de la colonne et l'entête « Food stamps » s'applique à tous les éléments de la ligne.

La première étape consiste à utiliser les bonnes balises pour réaliser un tableau accessible et de positionner ensuite les bons aria-labelledby dans les cellules correspondantes.

Edit Program Information

Program	Now	Past	Date
Food stamps	☐	☐	
TCA	☐	☐	
Medical	☐	☐	

Pour associer plusieurs entêtes à un élément du formulaire, il faut y renseigner l'id de chaque entête séparé par un espace. Par exemple, voici le code à utiliser pour associer un élément de formulaire à deux entêtes avec id=«now» et id=«foodstamps» :

aria-labelledby="now foodstamps"

Dans des tableaux encore plus complexes, il est possible d'utiliser 3 ou plus de 3 labels associés mais ce n'est pas une bonne pratique en matière d'usage d'un formulaire.

Voici le code source complet relatif au tableau illustré ci-dessus :

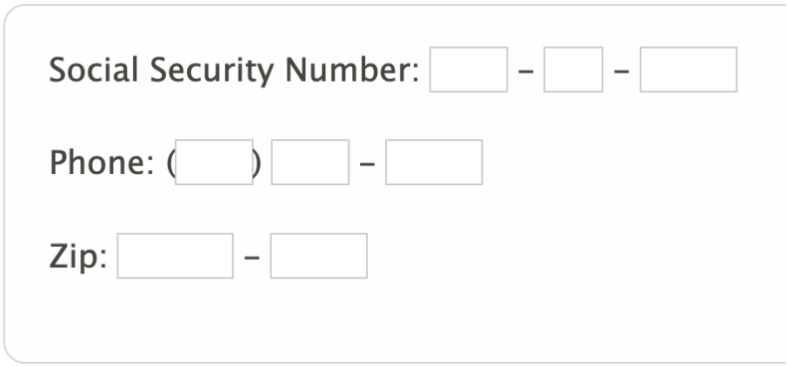
```
<table>
<caption>Edit Program Information</caption>
  <tr>
    <th scope="col"><span id="program">Program</span></th>
    <th scope="col"><span id="now">Now</span></th>
    <th scope="col"><span id="past">Past</span></th>
    <th scope="col"><span id="date">Date</span></th>
  </tr>
  <tr>
    <th scope="row"><span id="foodstamps">Food stamps</span></th>
    <td><input type="checkbox" name="checkbox" aria-labelledby="now foodstamps"></td>
    <td><input type="checkbox" name="checkbox2" aria-labelledby="past foodstamps"></td>
    <td><input type="text" name="textfield" aria-labelledby="date foodstamps"></td>
  </tr>
  <tr>
    <th scope="row"><span id="tca">TCA</span></th>
    <td><input type="checkbox" name="checkbox3" aria-labelledby="now tca"></td>
    <td><input type="checkbox" name="checkbox4" aria-labelledby="past tca"></td>
    <td><input type="text" name="textfield2" aria-labelledby="date tca"></td>
  </tr>
  <tr>
    <th scope="row"><span id="medical">Medical</span></th>
    <td><input type="checkbox" name="checkbox5" aria-labelledby="now medical"></td>
    <td><input type="checkbox" name="checkbox6" aria-labelledby="past medical"></td>
    <td><input type="text" name="textfield3" aria-labelledby="date medical"></td>
  </tr>
</table>
```

Attention !!! Les labels d'entête doivent être renseignés à l'intérieur de balises qui contiennent l'id. Si l'id est positionné sur le <th>, certains lecteurs d'écran risquent de ne pas lire correctement le label en tabulant sur l'élément du formulaire.

h) Un label pour plusieurs champs

Un label utilisé pour plusieurs éléments doit être techniquement associé à chacun d'eux.

Certains champs de formulaires sont découpés en plusieurs parties pour renseigner un numéro composé comme un numéro de compte, de sécurité sociale, de téléphone…



Il existe plusieurs techniques pour rédiger des labels accessibles dans ces situations :

1. Combiner les champs dans un seul champ pour éviter de multiplier les labels
2. Cacher les labels en CSS tout en les laissant lisibles par les lecteurs d'écrans. Cette technique utilise les <label>, est très robuste mais moins élégante que la technique des aria-labels.
3. Utiliser aria-label pour fournir des labels cachés en gardant un seul label qui permet à l'utilisateur de cliquer sur l'étiquette pour positionner le focus dans le champ.
4. Utiliser un <fieldset> pour le label visible et du texte caché pour les autres labels.

Bon exemple : Des champs multiples combinés en un seul avec un seul label

<label for="ssn6">Social Security Number:</label> <input type="text" id="ssn6">

Bon exemple : Des champs multiples avec un seul label et des labels cachés en CSS

<label for="ssn1b">Social Security Number first three digits:

</label>

<input type="text" size="3" maxlength="3" id="ssn1b"> -

<label for="ssn2b" class="offscreen">Second two digits:</label>

<input type="text" size="2" maxlength="2" id="ssn2b"> -

<label for="ssn3b" class="offscreen">Last four digits:</label>

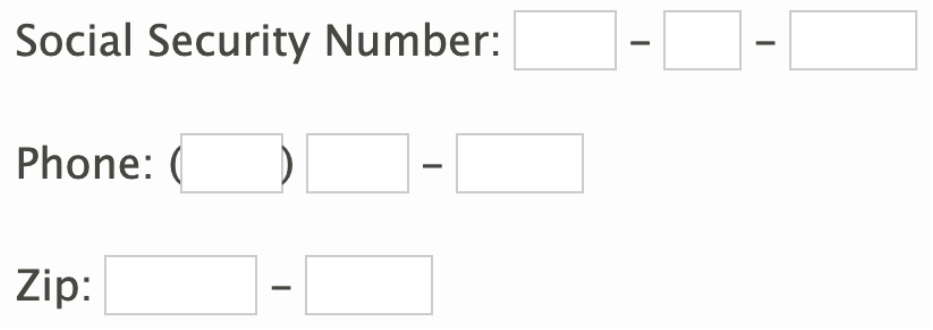
<input type="text" size="4" maxlength="4" id="ssn3b">

La class « offscreen » positionne le contenu à -9999px top et left par exemple.

Quelques notes sur cette méthode :

- Cette méthode est davantage rétro-compatible avec les anciens navigateurs et lecteurs d'écrans que la méthode des aria-labels (voir exemple suivant).
- Dans cette méthode, le lecteur d'écran ne répétera pas deux fois le même contenu donc la lecture est plus confortable et fluide.

Bon exemple : Des champs multiples avec un seul label visible et aria-label pour cacher des libellés.




```
<label for="ssn1">Social Security Number:</label>

<input type="text" size="3" maxlength="3" id="ssn1" aria-label="Social Security Number first three digits"> -

<input type="text" size="2" maxlength="2" aria-label="middle two digits"> -

<input type="text" size="4" maxlength="4" aria-label="last four digits">

</p>

<p>

  <label for="phone1">Phone:</label>

  (<input type="text" size="3" maxlength="3" id="phone1" aria-label="Phone Number area code">)

  <input type="text" size="3" maxlength="3" aria-label="second three digits"> -

  <input type="text" size="4" maxlength="4" aria-label="last four digits">

</p>

<p>

  <label for="zip">Zip:</label>

  <input type="text" size="5" maxlength="5" id="zip" aria-label="Zip code"> -

  <input type="text" size="4" maxlength="4" aria-label="extended zip code">

</p>
```

Quelques points d'attention relatifs à ce code :

- Le premier champ est lié à un label visible <label> en utilisant la combinaison id et for. Cette technique permet à tous les utilisateurs de positionner le focus dans le champ en cliquant sur l'étiquette (problèmes de dextérité, tremblements, vue basse…).
- Le premier champ contient un attribut aria-label en plus du <label> pour préciser aux utilisateurs de lecteur d'écran que ce champ attend les 3 premiers chiffres du numéro

du « Social Security Number ». Sans cette technique l'internaute renseignera l'ensemble des chiffres ce qui générera une erreur (maxlength = « 3 »).

- Au moment d'écrire ces lignes, JAWS et NVDA ne liront que la valeur du aria-label alors que VoiceOver lira les valeurs du aria-label et du label (ce qui n'est pas idéal).

Bon exemple : Des champs multiples avec un seul label visible et l'utilisation de <fieldset>

<fieldset>

<legend>Social Security Number</legend>

<label for="ssn10" class="offscreen">First three digits</label>

<input type="text" size="3" maxlength="3" id="ssn10"> -

<label for="ssn20" class="offscreen">Second two digits:</label>

<input type="text" size="2" maxlength="2" id="ssn20"> -

<label for="ssn30" class="offscreen">Last four digits:</label>

<input type="text" size="4" maxlength="4" id="ssn30">

</fieldset>

La class « offscreen » positionne le contenu à -9999px top et left par exemple.

Notes relatives à cette méthode :

- Peut également être utilisée avec aria-label comme vu précédemment
- Cette solution fonctionne très bien avec tous les lecteurs d'écran et est rétro-compatible
- Un inconvénient réside dans le fait que le <fieldset> et <legend> peuvent être plus compliqués à styler en comparaison avec du texte normal. Les navigateurs récents permettent de désactiver le border du <fieldset> et de positionner <legend> à la gauche du champ même si cela demande un peu plus de travail au designer. Les anciens navigateurs sont moins compatibles sur ce point.
- Autre inconvénient : Certains lecteurs d'écran (JAWS) répètent le texte de <legend> dans tous les champs qu'il contient. Dans le cas présent, l'utilisateur entendra « Social Security Number » dans les 3 champs. Ce n'est toutefois pas bloquant sauf si le texte est vraiment long et cet inconfort peut être facilement évité en tabulant directement après avoir entendu les informations importantes ou appuyer sur la touche (CTRL) pour stopper la lecture.

Labels de groupes de champs de formulaires

Les exigences relatives aux labels des champs de formulaires sont également valables pour les groupes de champs d'un formulaire pour permettre aux utilisateurs de les identifier facilement et de ne pas commettre d'erreurs. C'est par exemple le cas d'un groupe de boutons radio.

Les concepts vus pour les labels vont donc s'appliquer également dans les regroupements de champs (sémantique, texte explicite, proximité et visibilité).

a) Sémantique des labels de groupes

Les labels doivent être techniquement associés à leur regroupement s'ils sont insuffisants pris isolément.

Lorsqu'un regroupement de champs partagent le même label, le regroupement et son label doivent être sémantiquement explicites pour que l'utilisateur d'un lecteur d'écran puisse comprendre le lien entre les champs du formulaire, leur label associé et le groupement auquel ils appartiennent.

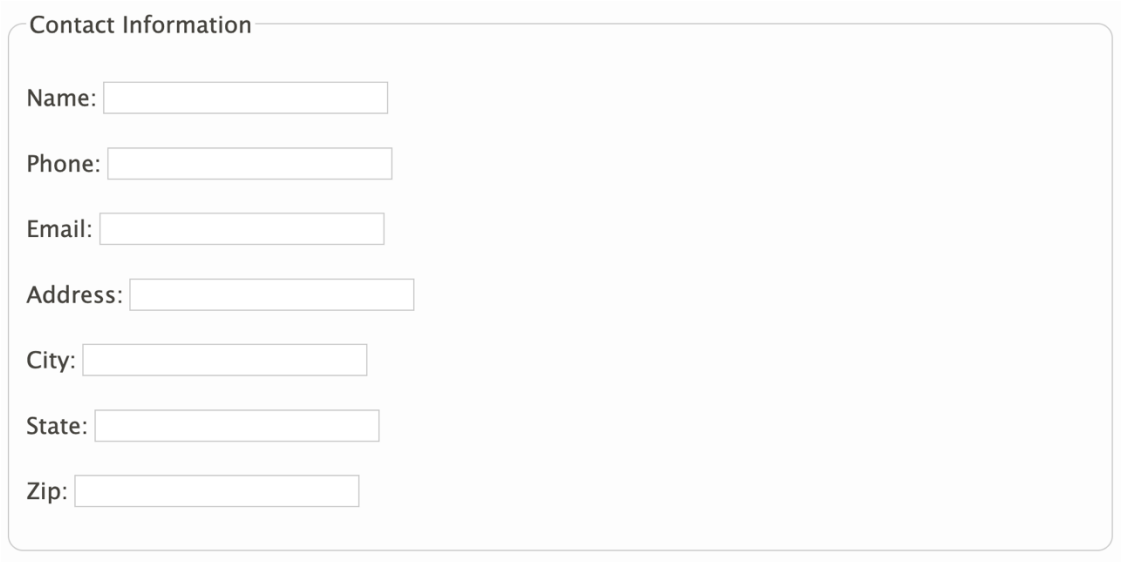
Le groupement de composants est très important dans le cas des boutons radio et des cases à cocher (radio buttons et checkboxes). En effet, même si le champ est correctement associé à son label, l'utilisateur ne pourra pas, par exemple, faire le lien entre un « oui » ou « non » s'il ne connaît pas la question à laquelle la réponse se rapporte.

La même logique est valable pour les champs qui font partie d'une même famille comme dans le cas des adresses d'envoi et de facturation.

Le groupement de champs d'un formulaire peut se faire de deux manières :

- La solution de base HTML en utilisant la balise <fieldset> pour regrouper les éléments et la balise <legend> pour y associer sémantiquement un label.
- La solution ARIA où les champs sont liés dans une région qui contient un role « group » et dont le nom est donné par un attribut aria-labelledby.

Bon exemple : Les éléments reliés entre eux avec <fieldset> et <legend>




```
<form>
  <fieldset>
    <legend>Contact Information</legend>
    <p><label for="name6044">Name: </label> <input type="text" id="name6044"></p>
    <p><label for="phone6044">Phone: </label> <input type="text" id="phone6044"></p>
    <p><label for="email6044">Email: </label> <input type="text" id="email6044"></p>
```

```
<p><label for="address6044">Address: </label> <input type="text" id="address6044"></p>
<p><label for="city6044">City: </label> <input type="text" id="city6044"></p>
<p><label for="state6044">State: </label> <input type="text" id="state6044"></p>
<p><label for="zip6044">Zip: </label> <input type="text" id="zip6044"></p>
</fieldset>
</form>
```

La balise <fieldset> regroupe les éléments d'une même famille entre eux visuellement, grâce à une bordure, et sémantiquement ce qui permet aux lecteurs d'écran de nommer le groupement aux utilisateurs.

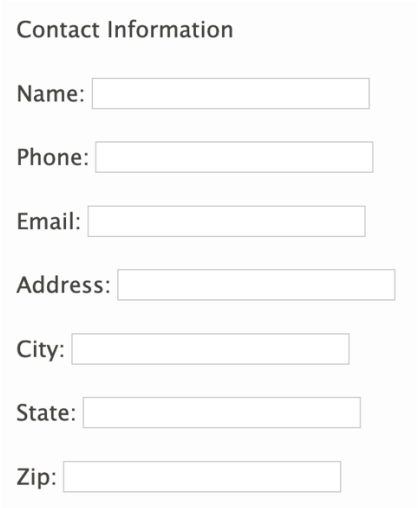
Certains lecteurs d'écran, comme JAWS et VoiceOver, lisent le texte contenu dans la <legend> en plus du nom de chaque champ du regroupement concerné. La plupart lisent d'abord la <legend> alors que VoiceOver lit d'abord le label du champ suivi de la légende du fieldset.

Bon exemple : Les éléments reliés entre eux avec ARIA

Les champs d'un même groupement sont placés dans un container, comme par exemple une , qui contient un rôle ARIA « group » et un attribut aria-labelledby associé à l'id d'un élément qui contient le texte de la légende.

Remarques :

- Cette technique ne dessine pas de bordure au container. Or, un regroupement visuel des éléments d'une même famille est recommandé. Nous recommandons de l'ajouter en CSS.
- A l'heure d'écrire ces lignes, toutes les versions de VoiceOver ne supportent pas le role= « group ». Le premier exemple est donc recommandé.



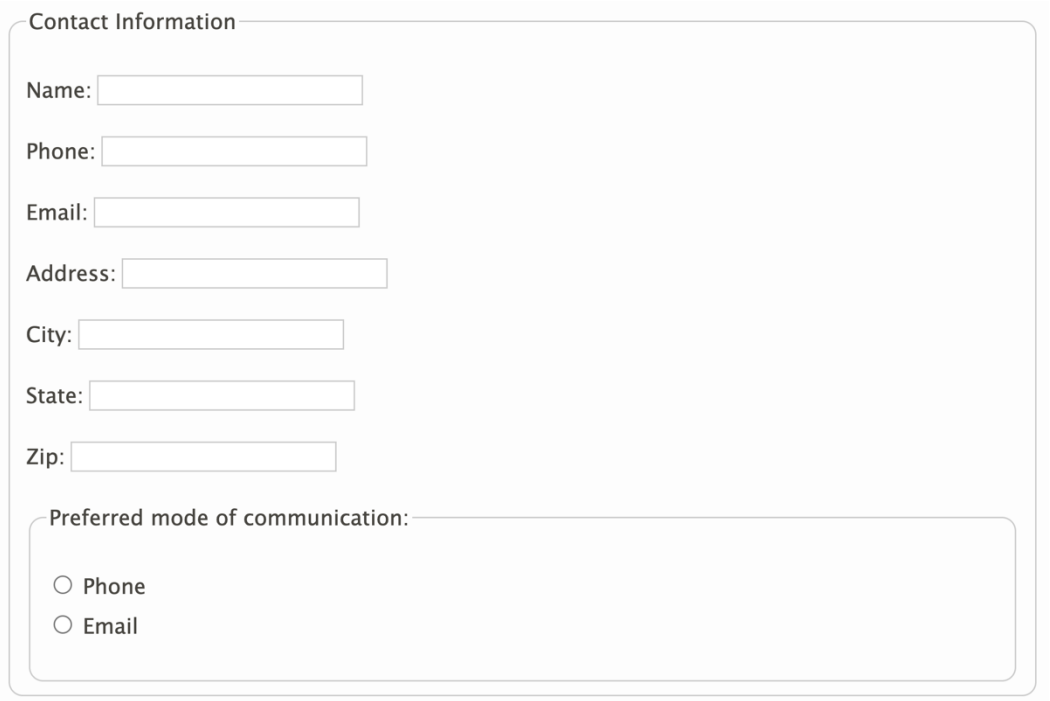

```
<p id="grouplabel9961">Contact Information</p>
<div role="group" aria-labelledby="grouplabel9961">
<p><label for="name">Name: </label> <input type="text" id="name"></p>
```

```
<p><label for="phone">Phone: </label> <input type="text" id="phone"></p>
<p><label for="email">Email: </label> <input type="text" id="email"></p>
<p><label for="address">Address: </label> <input type="text" id="address"></p>
<p><label for="city">City: </label> <input type="text" id="city"></p>
<p><label for="state">State: </label> <input type="text" id="state"></p>
<p><label for="zip">Zip: </label> <input type="text" id="zip"></p>
</div>
```

Bon exemple : Groupements d'éléments de formulaire de la même famille dans des <fieldset> imbriqués

Les <fieldset> imbriqués sont autorisés pour créer des regroupements dans un groupe.

Note : Les anciennes versions des navigateurs tentent de lire la légende du <fieldset> englobant puis celle du <fieldset> interne sur chaque élément du formulaire rendant ainsi la lecture redondante et moins confortable. Les navigateurs plus récents ont amélioré leur fonctionnement en lisant uniquement la légende du <fieldset> interne.




```
<fieldset>
 <legend>Contact Information</legend>
 <p><label for="name6045">Name: </label><input type="text" id="name6045"></p>
 <p><label for="phone6045">Phone: </label><input type="text" id="phone6045"></p>
 <p><label for="email6045">Email: </label><input type="text" id="email6045"></p>
 <p><label for="address6045">Address: </label><input type="text" id="address6045"></p>
 <p><label for="city6045">City: </label><input type="text" id="city6045"></p>
 <p><label for="state6045">State: </label><input type="text" id="state6045"></p>
```

```
<p><label for="zip6045">Zip: </label><input type="text" id="zip6045"></p>
<div class="mode-radio">
  <fieldset>
    <legend>Preferred mode of communication:</legend>
    <p><input type="radio" name="mode" id="mode-phone6045">
      <label for="mode-phone6045">Phone</label><br>
    <input type="radio" name="mode" id="mode-email6045">
      <label for="mode-email6045">Email</label></p>
  </fieldset>
</div>
</fieldset>
```

Mauvais exemple : Faux regroupement

Les éléments de ce formulaire sont regroupés visuellement en CSS sans être associés sémantiquement. L'utilisateur de lecteur d'écran ne comprend donc pas le lien entre les éléments.

Code HTML:

```
<div class="group1">
  <span><strong>What is your preferred method of contact?</strong></span>
  <div>
    <input type="radio" name="contact" value="email" id="email9017">
      <label for="email9017">Email</label><br>
    <input type="radio" name="contact" value="phone" id="phone9017">
      <label for="phone9017">Phone</label><br>
    <input type="radio" name="contact" value="mail" id="mail9017">
      <label for="mail9017">Mail</label>
  </div>
</div>
```

Code CSS:

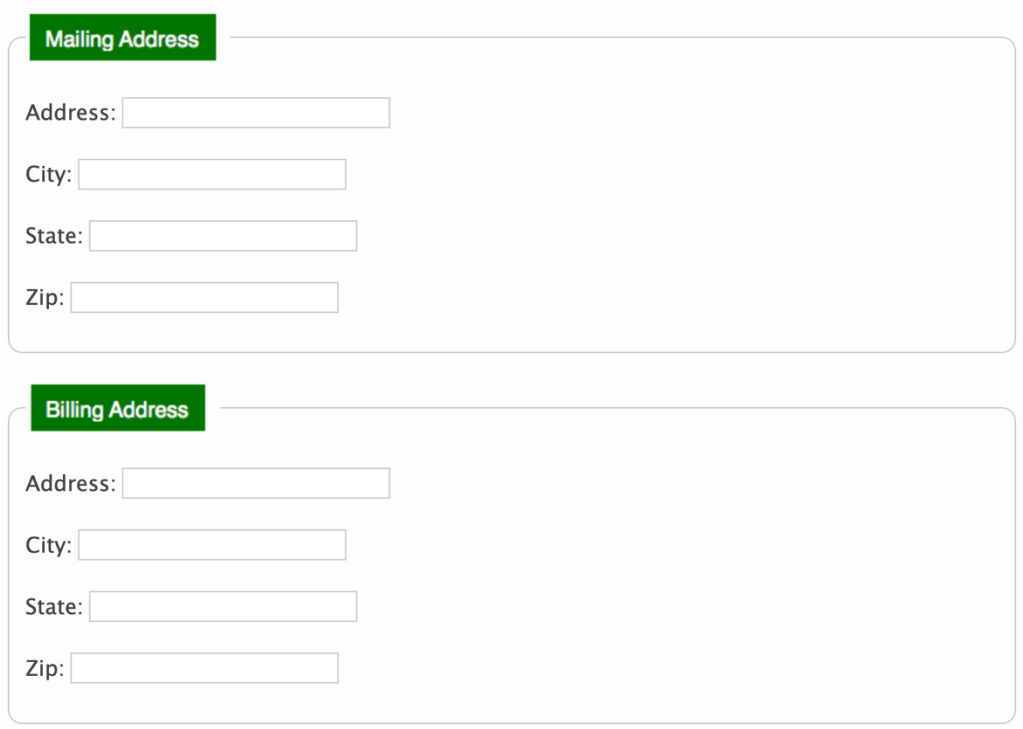
```
.group1 {
border: 1px solid grey;
border-radius: 10px;
padding: 10px;
}
```

Les labels des regroupements doivent être techniquement déterminables

Les spécifications WCAG et HTML n'obligent pas l'ajout systématique d'une balise <legend> à chaque <fieldset>. Cependant, lorsqu'un <fieldset> regroupe des éléments, il faut lui donner une légende qui sera utilisée par les lecteurs d'écrans.

Les deux techniques qui permettent d'associer une légende à un regroupement sont la balise <legend> ou l'attribut aria-labelledby en cas d'usage d'ARIA.

Mauvais exemple : Un regroupement sans label



Bien que le regroupement soit réalisé à l'aide d'un <fieldset> légendé par <legend>, l'image utilisée dans la légende ne contient pas d'alternative textuelle qui décrit l'image utilisée. Par ailleurs, les personnes malvoyantes qui doivent zoomer leur écran, modifier la police, changer sa couleur ou son contraste… risquent également de se trouver un situation de handicap avec cette méthode.

```
<div>
  <fieldset>
    <legend><img src="assets/images/module-forms/maddress.png"></legend>
    <p><label for="mailaddress9014">Address: </label>
      <input type="text" id="mailaddress9014"></p>
    <p><label for="mailcity9014">City: </label>
```

```
        <input type="text" id="mailcity9014"></p>
      <p><label for="mailstate9014">State: </label>
        <input type="text" id="mailstate9014"></p>
      <p><label for="mailzip9014">Zip: </label>
        <input type="text" id="mailzip9014"></p>
  </fieldset>
</div>
<div>
  <fieldset>
    <legend><img src="assets/images/module-forms/baddress.png"></legend>
    <p><label for="billaddress9014">Address: </label>
      <input type="text" id="billaddress9014"></p>
    <p><label for="billcity9014">City: </label>
      <input type="text" id="billcity9014"></p>
    <p><label for="billstate9014">State: </label>
      <input type="text" id="billstate9014"></p>
    <p><label for="billzip9014">Zip: </label>
      <input type="text" id="billzip9014"></p>
  </fieldset>
</div>
```

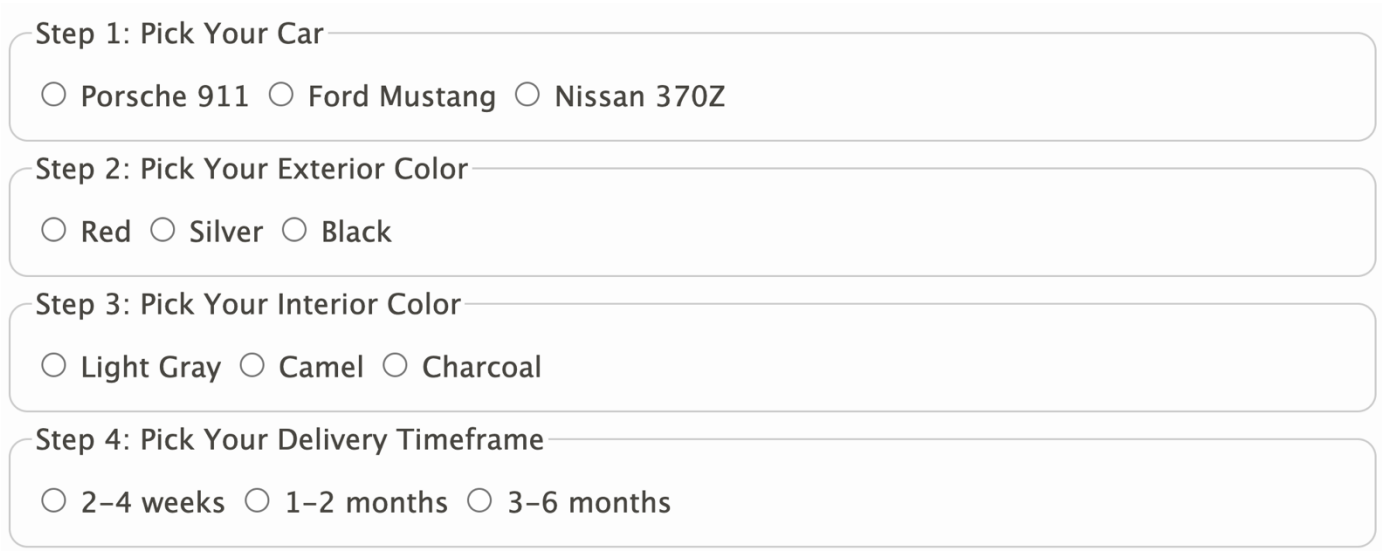
b) Labels de groupements explicites

Les labels de regroupements doivent être explicites

Les utilisateurs doivent savoir exactement les données à compléter dans le formulaire.

Bon exemple : Un texte de légende explicite

Les champs ci-dessous sont répartis dans 4 <fieldset> associés à une <legend> qui fournit le numéro de l'étape et l'information demandée.

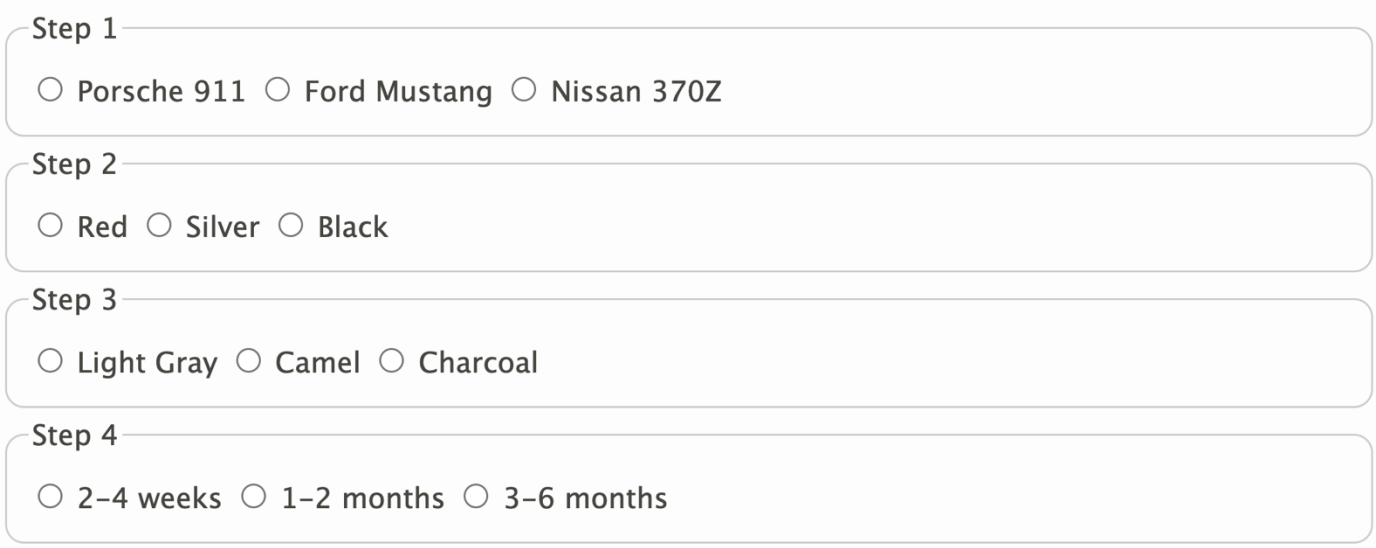



```
<fieldset>
  <legend>Step 1: Pick Your Car</legend>
  <input type="radio" id="porsche9996" name="car"> <label for="porsche9996">Porsche 911</label>
  <input type="radio" id="mustang9996" name="car"> <label for="mustang9996">Ford Mustang</label>
  <input type="radio" id="nissan9996" name="car"> <label for="nissan9996">Nissan 370Z</label>
</fieldset>
<fieldset>
  <legend>Step 2: Pick Your Exterior Color</legend>
  <input type="radio" id="red9996" name="ecolor"> <label for="red9996">Red</label>
  <input type="radio" id="silver9996" name="ecolor"> <label for="silver9996">Silver</label>
  <input type="radio" id="black9996" name="ecolor"> <label for="black9996">Black</label>
</fieldset>
<fieldset>
  <legend>Step 3: Pick Your Interior Color</legend>
  <input type="radio" id="gray9996" name="icolor"> <label for="gray9996">Light Gray</label>
  <input type="radio" id="camel9996" name="icolor"> <label for="camel9996">Camel</label>
  <input type="radio" id="char9996" name="icolor"> <label for="char9996">Charcoal</label>
</fieldset>
<fieldset>
  <legend>Step 4: Pick Your Delivery Timeframe</legend>
```

```
<input type="radio" id="short9996" name="delivery"> <label for="short9996">2-4 weeks</label>

<input type="radio" id="medium9996" name="delivery"> <label for="medium9996">1-2 months</label>

<input type="radio" id="long9996" name="delivery"> <label for="long9996">3-6 months</label>

</fieldset>
```

Mauvais exemple : Un texte de légende ambigu

Dans le même exemple que le précédent, seul le numéro de l'étape est renseigné. L'utilisateur ne connaît pas l'information demandée.




```
<fieldset>
  <legend>Step 1</legend>
  <input type="radio" id="porsche9997" name="car2">
    <label for="porsche9997">Porsche 911</label>
  <input type="radio" id="mustang9997" name="car2">
    <label for="mustang9997">Ford Mustang</label>
  <input type="radio" id="nissan9997" name="car2">
    <label for="nissan9997">Nissan 370Z</label>
</fieldset>
<fieldset>
  <legend>Step 2</legend>
  <input type="radio" id="red9997" name="ecolor2">
    <label for="red9997">Red</label>
  <input type="radio" id="silver9997" name="ecolor2">
```

```
    <label for="silver9997">Silver</label>
  <input type="radio" id="black9997" name="ecolor2">
    <label for="black9997">Black</label>
</fieldset>
<fieldset>
  <legend>Step 3</legend>
  <input type="radio" id="gray9997" name="icolor2">
    <label for="gray9997">Light Gray</label>
  <input type="radio" id="camel9997" name="icolor2">
    <label for="camel9997">Camel</label>
  <input type="radio" id="char9997" name="icolor2">
    <label for="char9997">Charcoal</label>
</fieldset>
<fieldset>
 <legend>Step 4</legend>
  <input type="radio" id="short9997" name="delivery2">
    <label for="short9997">2-4 weeks</label>
  <input type="radio" id="medium9997" name="delivery2">
    <label for="medium9997">1-2 months</label>
  <input type="radio" id="long9997" name="delivery2">
    <label for="long9997">3-6 months</label>
</fieldset>
```

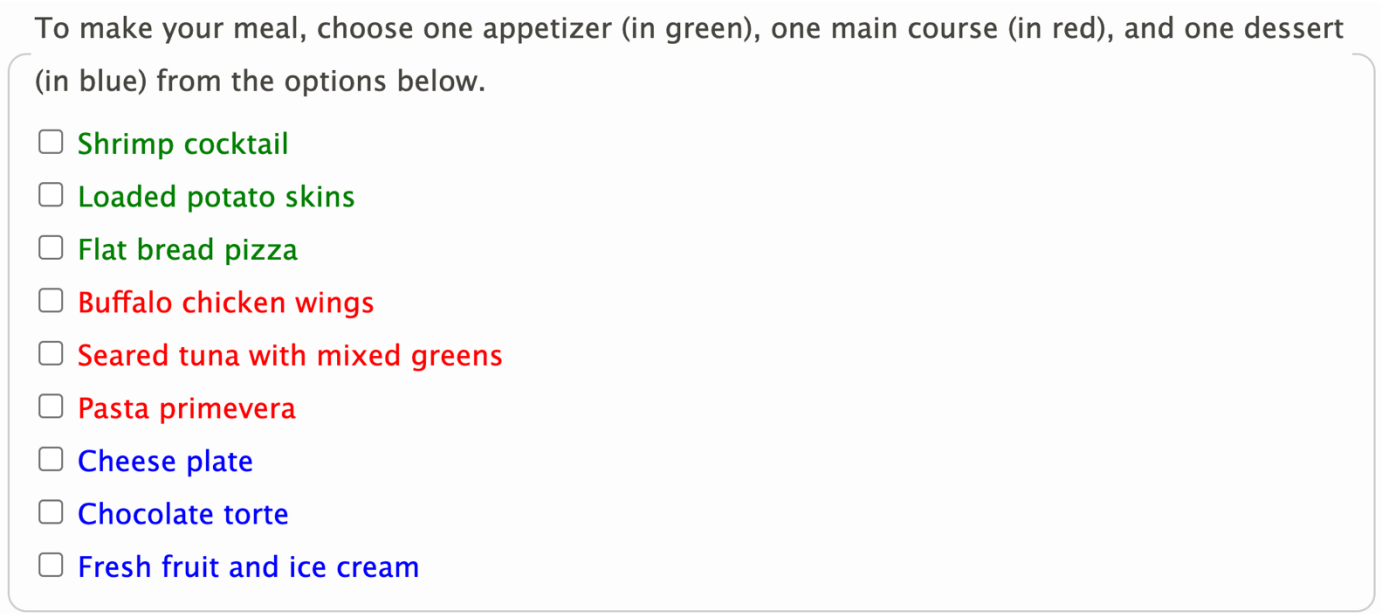
Les labels des regroupements ne peuvent fournir des informations uniquement par la couleur, l'orientation, un son, la position ou la forme.

Donner une information uniquement par une caractéristique sensorielle peut exclure certaines personnes de par leurs besoins spécifiques. Il est nécessaire de garder quelques points à l'esprit :

- Si la couleur est utilisée dans un label pour fournir une information, s'assurer que la couleur soit doublée par un autre facteur comme du texte ou un symbole reconnu et implémenté de manière accessible.
- Le contraste des textes, symboles et icônes doivent respecter les ratios d'accessibilité.
- Toute information visuelle doit être techniquement utilisable par les utilisateurs de synthèses vocales.

Mauvais exemple : Des libellés qui ne donnent des informations que par des caractéristiques sensorielles.

Dans cet exemple, le développeur part du principe que l'utilisateur peut distinguer les couleurs pour choisir son menu. Les personnes daltoniennes et les utilisateurs de synthèses vocales ne pourront répondre de manière appropriée à ce formulaire.




```
<fieldset>
  <legend>To make your meal, choose one appetizer (in green), one main course (in red), and one dessert (in blue) from the options below.</legend>
  <input type="checkbox" name="meal" value="shrimp" id="shrimp"> <label for="shrimp" style="color: green">Shrimp cocktail</label><br>
  <input type="checkbox" name="meal" value="potato" id="potato"> <label for="potato" style="color: green">Loaded potato skins</label><br>
  <input type="checkbox" name="meal" value="pizza" id="pizza"> <label for="pizza" style="color: green">Flat bread pizza</label><br>
  <input type="checkbox" name="meal" value="wings" id="wings"> <label for="wings" style="color: red">Buffalo chicken wings</label><br>
  <input type="checkbox" name="meal" value="tuna" id="tuna"> <label for="tuna" style="color: red">Seared tuna with mixed greens</label><br>
  <input type="checkbox" name="meal" value="pasta" id="pasta"> <label for="pasta" style="color: red">Pasta primevera</label><br>
  <input type="checkbox" name="meal" value="cheese" id="cheese"> <label for="cheese" style="color: blue">Cheese plate</label><br>
  <input type="checkbox" name="meal" value="torte" id="torte"> <label for="torte" style="color: blue">Chocolate torte</label><br>
  <input type="checkbox" name="meal" value="fruit" id="fruit"> <label for="fruit" style="color: blue">Fresh fruit and ice cream</label><br>
```

</fieldset>

Bon exemple : Des libellés qui donnent de l'information par la couleur et par une autre technique.

Dans cet exemple, les différents types de plats sont séparés par des <fieldset> associés à leur <legend> explicite.

<fieldset>

<legend>To make your meal, choose one appetizer, one main course, and one dessert from the options below.</legend>

<fieldset>

<legend>Appetizers</legend>

<input type="checkbox" name="meal" value="shrimp" id="shrimp1"> <label for="shrimp1" style="color: green">Shrimp cocktail</label>

<input type="checkbox" name="meal" value="potato" id="potato1"> <label for="potato1" style="color: green">Loaded potato skins</label>

<input type="checkbox" name="meal" value="pizza" id="pizza1"> <label for="pizza1" style="color: green">Flat bread pizza</label>

</fieldset>

<fieldset>

<legend>Main Courses</legend>

```
<input type="checkbox" name="meal" value="wings" id="wings1"> <label for="wings1" style="color: red">Buffalo chicken wings</label><br>

<input type="checkbox" name="meal" value="tuna" id="tuna1"> <label for="tuna1" style="color: red">Seared tuna with mixed greens</label><br>

<input type="checkbox" name="meal" value="pasta" id="pasta1"> <label for="pasta1" style="color: red">Pasta primevera</label>

</fieldset>

<fieldset>

<legend>Desserts</legend>

<input type="checkbox" name="meal" value="cheese" id="cheese1"> <label for="cheese1" style="color: blue">Cheese plate</label><br>

<input type="checkbox" name="meal" value="torte" id="torte1"> <label for="torte1" style="color: blue">Chocolate torte</label><br>

<input type="checkbox" name="meal" value="fruit" id="fruit1"> <label for="fruit1" style="color: blue">Fresh fruit and ice cream</label>

</fieldset>

</fieldset>
```

c) Proximité des labels de regroupements

Les labels de regroupements devraient être visuellement proches des éléments auxquels ils se rapportent

Les utilisateurs d'un zoom doivent pouvoir faire le lien visuellement entre le libellé et les éléments liés.

Mauvais exemple d'un label de regroupement loin des champs concernés

Une grande image interfère visuellement entre la légende et les checkboxes.

Choose your top three favorite holidays:

- [ ] Memorial Day
- [ ] 4th of July
- [ ] Labor Day
- [ ] Halloween
- [ ] Thanksgiving

```
<fieldset>

<legend>Choose your top three favorite holidays:</legend>

<p><img src="assets/images/module-forms/fireworks.png" height="196" width="625" alt=""></p>

  <input type="checkbox" name="holiday" id="memorial9142" value="Memorial"> <label for="memorial9142">Memorial Day</label><br>

  <input type="checkbox" name="holiday" id="july9142" value="July"> <label for="july9142">4th of July</label><br>

  <input type="checkbox" name="holiday" id="labor9142" value="Labor"> <label for="labor9142">Labor Day</label><br>

  <input type="checkbox" name="holiday" id="halloween9142" value=""> <label for="halloween9142">Halloween</label><br>

  <input type="checkbox" name="holiday" id="thanks9142" value=""> <label for="thanks9142">Thanksgiving</label><br>

</fieldset>
```

Bon exemple : le label est proche des champs du formulaire

Choose your top three favorite holidays:

- [ ] Memorial Day
- [ ] 4th of July
- [ ] Labor Day
- [ ] Halloween
- [ ] Thanksgiving

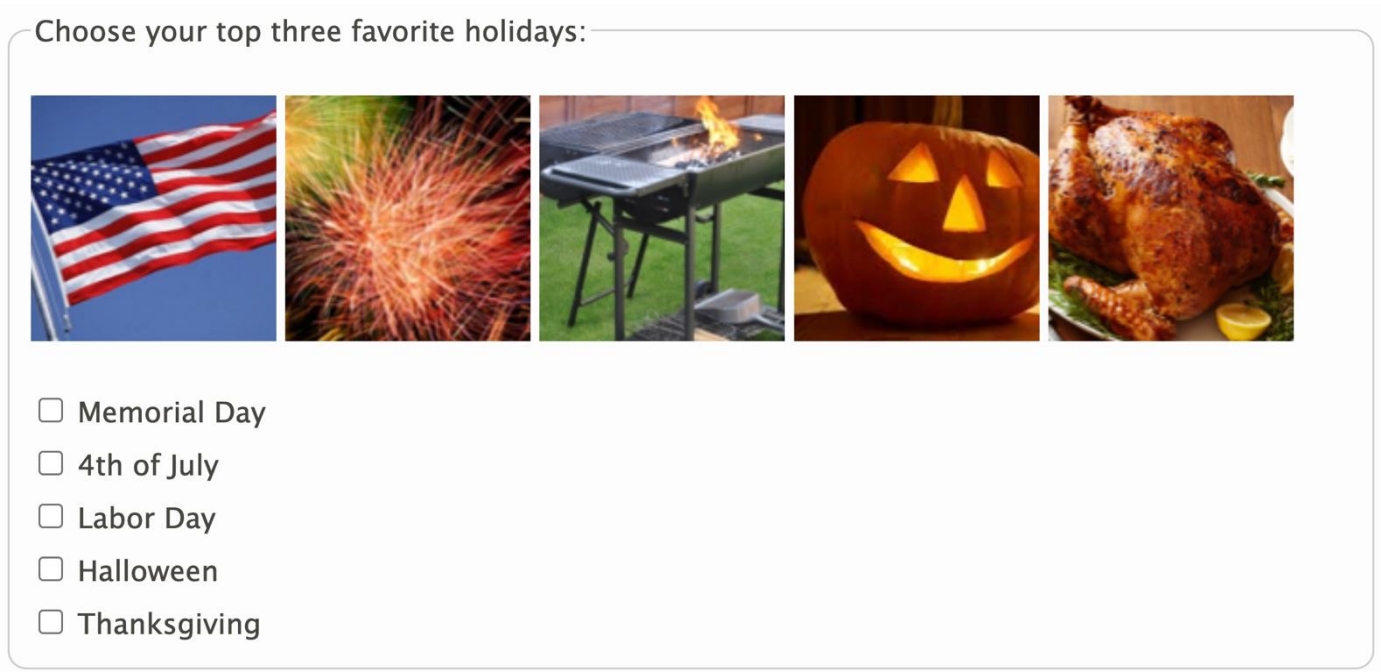
```
<p><img src="assets/images/module-forms/fireworks.png" height="196" width="625" alt=""></p>

<fieldset>

<legend>Choose your top three favorite holidays:</legend>

  <input type="checkbox" name="holiday" id="memorial9134" value="Memorial"> <label for="memorial9134">Memorial Day</label><br>

  <input type="checkbox" name="holiday" id="july9134" value="July"> <label for="july9134">4th of July</label><br>

  <input type="checkbox" name="holiday" id="labor9134" value="Labor"> <label for="labor9134">Labor Day</label><br>

  <input type="checkbox" name="holiday" id="halloween9134" value=""> <label for="halloween9134">Halloween</label><br>

  <input type="checkbox" name="holiday" id="thanks9134" value=""> <label for="thanks9134">Thanksgiving</label><br>

</fieldset>
```

Les labels de regroupements devraient être proches des éléments auxquels ils se rapportent dans le DOM

Les lecteurs d'écran lisent le contenu d'une page dans son ordre d'apparition dans le DOM. Afin de ne pas induire l'utilisateur en erreur, les labels de regroupement doivent être proches de leurs champs.

Mauvais exemple : le label de regroupement n'est pas proche des éléments dans le DOM

Ce formulaire est correctement structuré à l'aide d'un <fieldset> et d'un <legend> associé. En tabulant, l'utilisateur de lecteur d'écran bénéficiera d'une bonne expérience de surf. Cependant, si l'utilisateur utilise les flèches de navigation, il devra passer par les 5 images avant de lire les cases à cocher.




```
<fieldset>
<legend>Choose your top three favorite holidays:</legend>
<p><img src="memorial-sm.png" height="130" width="130" alt="Memorial Day">
<img src="fireworks-sm.png" height="130" width="130" alt="4th of July Fireworks">
<img src="bbq-sm.png" height="130" width="130" alt="Labor Day BBQ">
<img src="halloween-sm.png" height="130" width="130" alt="Halloween Pumpkin">
<img src="thanksgiving-sm.png" height="130" width="130" alt="Thanksgiving Turkey"></p>
   <input type="checkbox" name="holiday" id="memorial9129" value="Memorial">
      <label for="memorial9129">Memorial Day</label><br>
   <input type="checkbox" name="holiday" id="july9129" value="July">
      <label for="july9129">4th of July</label><br>
   <input type="checkbox" name="holiday" id="labor9129" value="Labor">
      <label for="labor9129">Labor Day</label><br>
   <input type="checkbox" name="holiday" id="halloween9129" value="Halloween">
      <label for="halloween9129">Halloween</label><br>
   <input type="checkbox" name="holiday" id="thanks9129" value="Thanksgiving">
```

```
    <label for="thanks9129">Thanksgiving</label><br>
</fieldset>
```

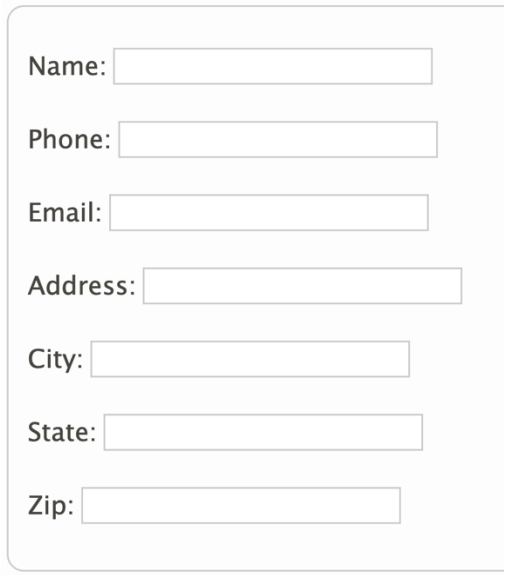
d) Visibilité des labels de regroupements

Les labels de regroupements doivent être visibles.

Les labels doivent toujours être visibles de manière à permettre à tous les utilisateurs de connaître les informations demandées dans les champs de formulaire.

Mauvais exemple : un texte de légende caché.

Ce formulaire utilise correctement un <legend> pour fournir de l'information aux utilisateurs de lecteurs d'écrans mais le <legend> est caché en CSS aux autres utilisateurs qui perdent le sens de cette partie du formulaire.

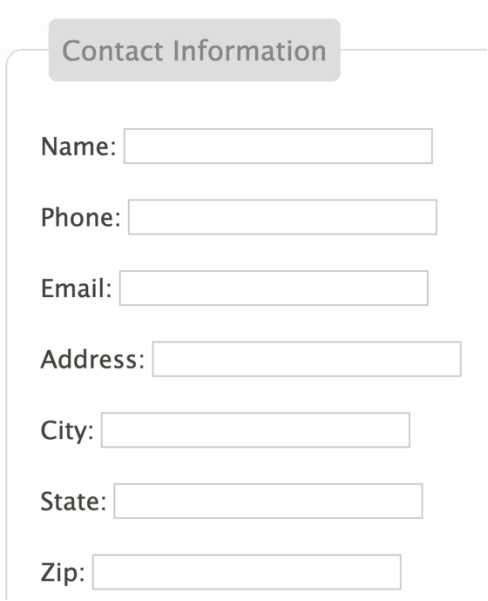



```
<form class="labelformat1">
 <fieldset class="fieldsetformat9968">
  <legend class="visually-hidden">Contact Information</legend>
  <p>
   <label for="name9968">Name: </label>
   <input type="text" id="name9968">
  </p>
  <p>
   <label for="phone9968">Phone: </label>
   <input type="text" id="phone9968">
  </p>
  <p>
   <label for="email9968">Email: </label>
   <input type="text" id="email9968">
```

```
  </p>
  <p>
    <label for="address9968">Address: </label>
    <input type="text" id="address9968">
  </p>
  <p>
    <label for="city9968">City: </label>
    <input type="text" id="city9968">
  </p>
  <p>
    <label for="state9968">State: </label>
    <input type="text" id="state9968">
  </p>
  <p>
    <label for="zip9968">Zip: </label>
    <input type="text" id="zip9968">
  </p>
 </fieldset>
</form>
```

Mauvais exemple : une légende mal contrastée

Ce formulaire est bien structuré mais le <legend> ne respecte pas les standards d'accessibilité en matière de contraste.



```
<form class="labelformat1">
  <fieldset class="fieldsetformat9081">
    <legend class="legendformat9081">Contact Information</legend>
    <p>
      <label for="id85_name3">Name: </label>
      <input type="text" id="id85_name3">
    </p>
    <p>
      <label for="id85_phone3">Phone: </label>
      <input type="text" id="id85_phone3">
    </p>
    <p>
      <label for="id85_email3">Email: </label>
      <input type="text" id="id85_email3">
    </p>
    <p>
      <label for="id85_address3">Address: </label>
      <input type="text" id="id85_address3">
    </p>
    <p>
      <label for="id85_city3">City: </label>
      <input type="text" id="id85_city3">
    </p>
    <p>
      <label for="id85_state3">State: </label>
      <input type="text" id="id85_state3">
    </p>
    <p>
      <label for="id85_zip3">Zip: </label>
      <input type="text" id="id85_zip3">
    </p>
  </fieldset>
```

</form>

Instructions et autres messages d'information

Un des meilleurs moyens d'améliorer l'accessibilité des formulaires est d'aider l'utilisateur à ne pas commettre d'erreurs. A cet effet, il est important de fournir à l'internaute des instructions claires le plus rapidement possible avant même qu'il n'entre de données. Les développeurs devront garder à l'esprit les points suivants :

- S'assurer que les labels et les instructions soient clairs et informatifs
- Rendre les instructions et messages d'information utilisables par les technologies d'assistance
- Indiquer clairement les restrictions pour les champs de formulaires
- Indiquer clairement les champs requis

a) Instructions pour les formulaires, les groupes et les sections

Les instructions pour les groupes ou sections devraient être techniquement associées au groupe.

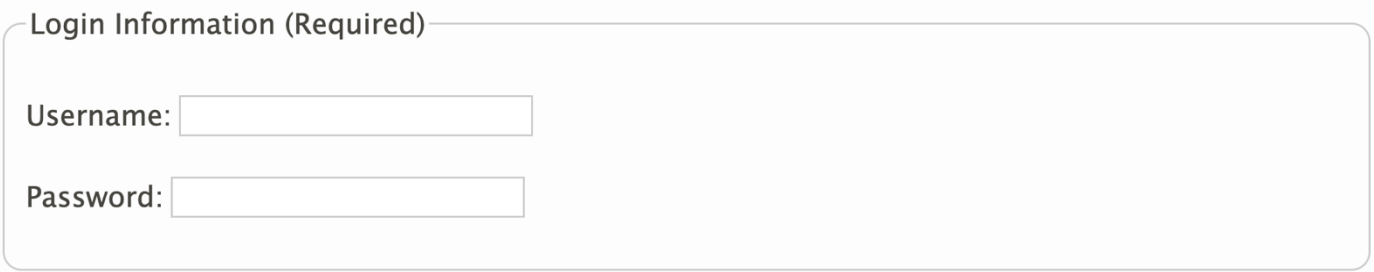
Renseigner du texte libre au milieu d'un formulaire sans l'associer à un label ou un champ peut empêcher l'utilisateur de l'entendre s'il parcourt le formulaire via la tabulation (seulement sur les éléments qui prennent le focus).

Il n'existe pas de technique native pour associer une description à un groupe. L'ajout d'aria-describedby au <fieldset> ou au <legend> ne fonctionne pas et il faut donc trouver un autre moyen.

- Option 1 : L'ajout des instructions au <legend>. Certains lecteurs d'écran lisent toutefois le texte de la légende au passage de chaque champ du formulaire associés. Cette option doit donc être réservée aux instructions courtes qui restent cohérentes avec chaque champ.
- Option 2 : Associer les instructions à un des champs u groupe (idéalement le premier) en utilisant aria-describedby. De cette manière, les instructions du groupe ne seront lues qu'une seule fois et lues par le lecteur d'écran lors d'une consultation via la tabulation.
- Option 3 : Ajouter les instructions avant le début du formulaire comme texte libre non associé et espérer que l'utilisateur lise le contenu du message avant d'entrer dans le formulaire. Cette solution contourne le problème et laisse la charge à l'utilisateur, ce qui n'est pas l'idéal même si beaucoup d'internautes lisent de texte de cette manière, paragraphe après paragraphe.

Bon exemple : Les instructions pour le groupe dans la <legend>

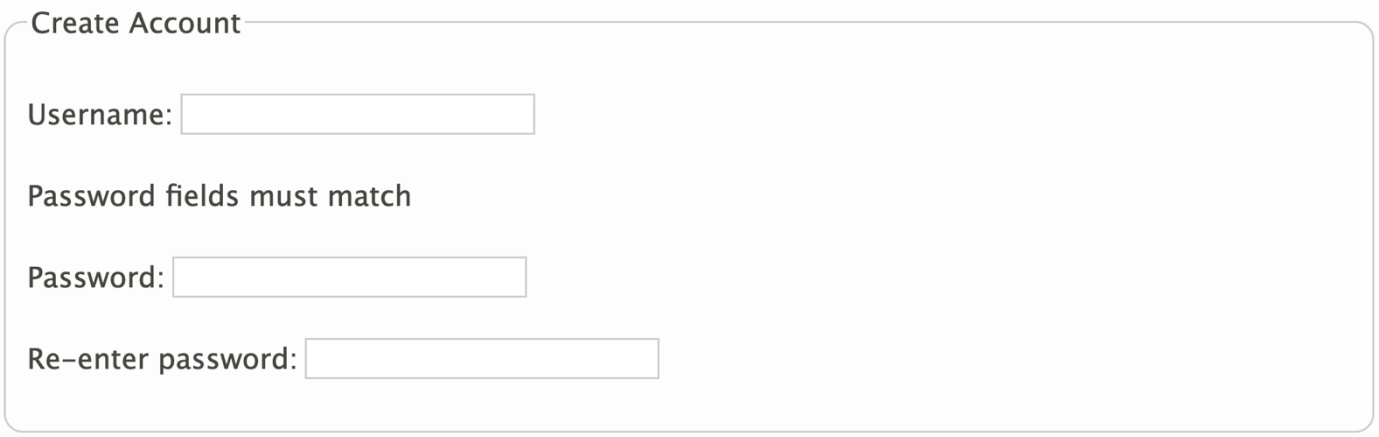
Le mot « Required » est indiqué dans la <legend> pour indiquer que tous les champs du <fieldset> sont obligatoires. Les lecteurs d'écran liront le texte de la <legend> lorsque l'utilisateur passe dans le premier champ.



```
<fieldset>
<legend>Login Information (Required)</legend>
  <label for="username">Username: </label>
  <input type="text" id="username"></p>
  <label for="password">Password: </label>
  <input type="password" id="password"></p>
</fieldset>
```

Bon exemple : Les instructions du groupe sont associées grâce à aria-describedby

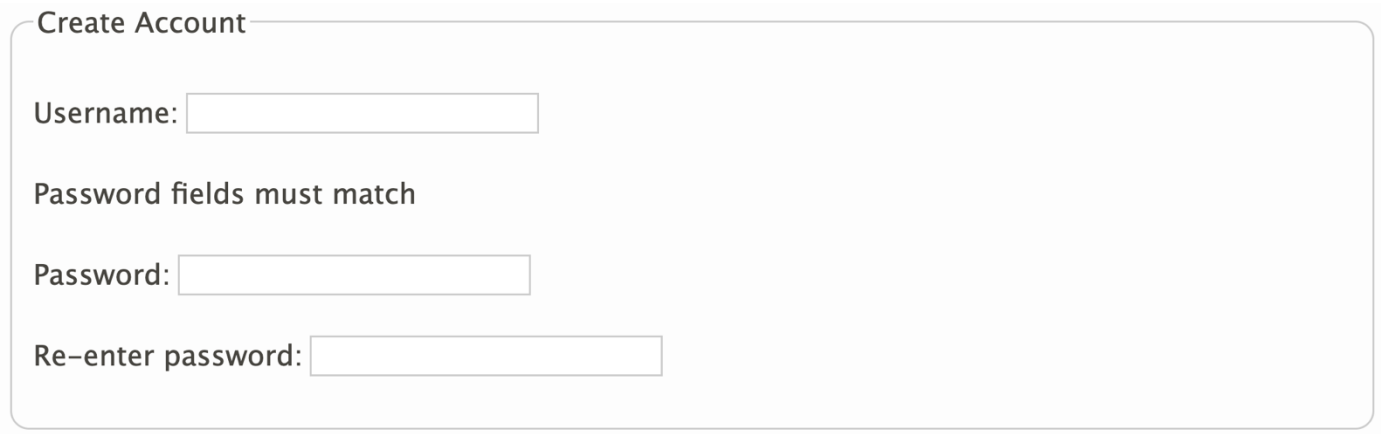
L'attribut aria-describedby est utilisé pour associer les instructions au premier champ « password » du formulaire.




```
<fieldset>
<legend>Create Account</legend>
  <p><label for="username">Username: </label>
    <input type="text" id="username"></p>
  <p id="mustmatch">Password fields must match</p>
  <p><label for="password">Password: </label>
    <input type="password" id="password" aria-describedby="mustmatch"></p>
  <p><label for="password2">Re-enter password: </label>
    <input type="password" id="password2"></p>
</fieldset>
```

Mauvais exemple : Les instructions sont dans un paragraphe non associé au groupe.

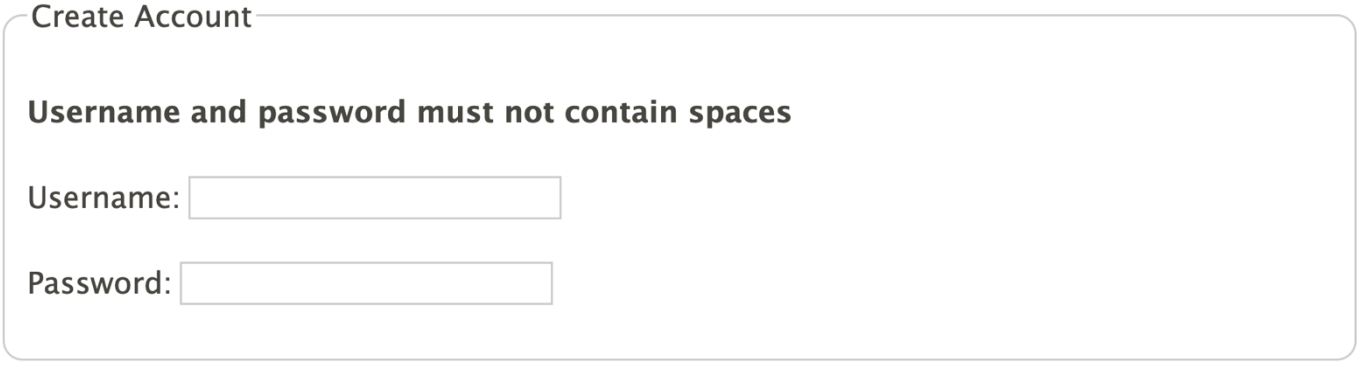
Le contenu du paragraphe <p> ne prend pas le focus. S'il n'est pas associé via, par exemple, aria-describedby, le texte ne sera pas entendu par l'utilisateur de lecteur d'écran.




```
<fieldset>
<legend>Create Account</legend>
  <p><label for="username">Username: </label>
    <input type="text" id="username"></p>
  <p>Password fields must match</p>
  <p><label for="password">Password: </label>
    <input type="password" id="password"></p>
  <p><label for="password2-">Re-enter password: </label>
    <input type="password" id="password2-"></p>
</fieldset>
```

Bon exemple : Des instructions associées à plusieurs champs du formulaire

Dans certains cas, il est utile d'associer un message à plusieurs champs mais pas à la totalité des champs du formulaire. Dans cet exemple, le message « Username et password ne peuvent pas contenir d'espaces » n'est valable que pour deux champs.




```
<fieldset>
<legend>Create Account</legend>
  <p><strong>Username and password
  <span id="mustnot">must not contain spaces</span></strong></p>
```

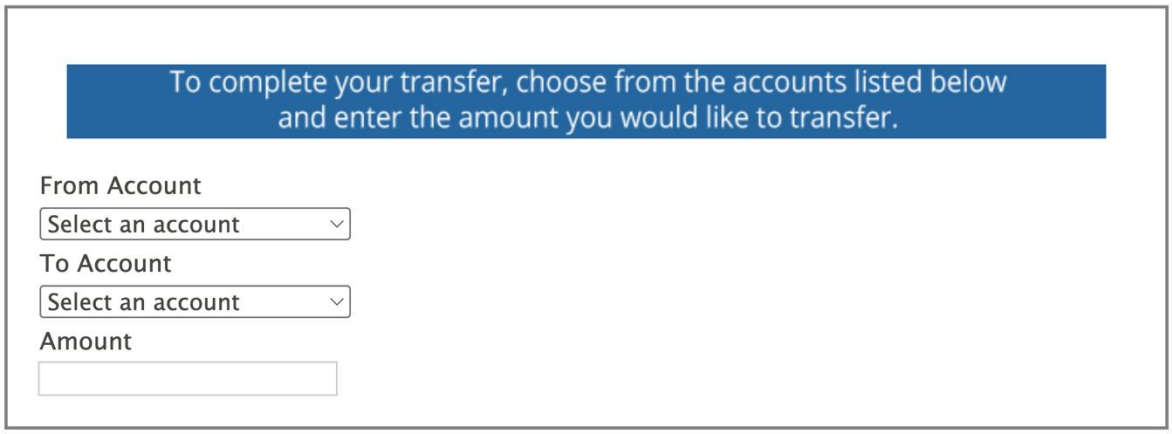
```
<p><label for="username">Username:</label>
  <input type="text" id="username" aria-describedby="mustnot"></p>
<p><label for="password">Password:</label>
  <input type="password" id="password" aria-describedby="mustnot"></p>
</fieldset>
```

Les instructions pour les groupes ou sections doivent être techniquement déterminables.

Les instructions doivent être renseignées sous forme de texte réel pour pouvoir être lues par tous les utilisateurs et les technologies d'assistance.

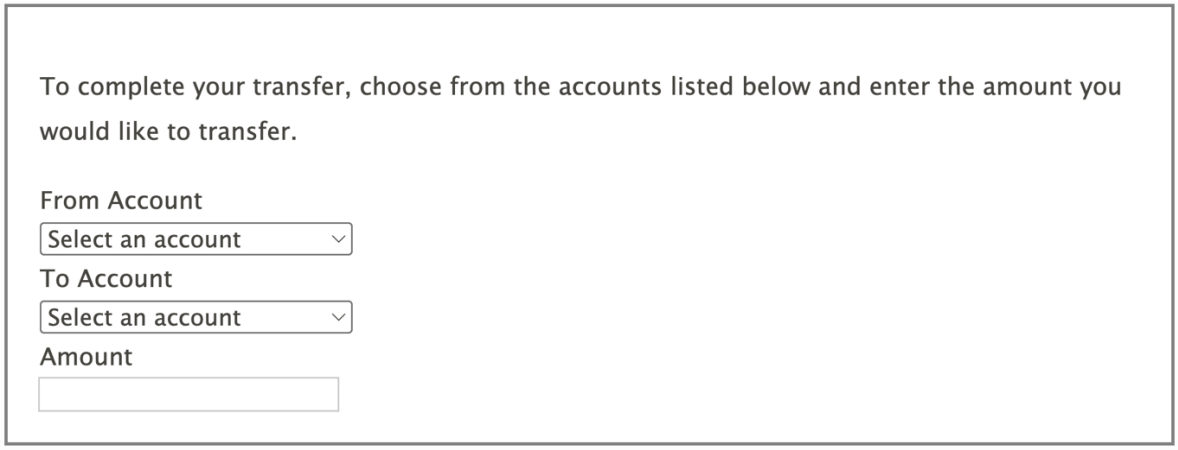
Mauvais exemple : Des instructions dans une image sans texte alternatif

Le texte d'instructions est présenté sous la forme d'une image sans attribut alt ou avec un attribut alt vide.



Remarque : Même si l'image contient une alternative correcte, une image-texte reste problématique pour les utilisateurs d'un zoom-text ou pour les internautes qui ont besoin de modifier l'affichage du texte (couleur, taille, police, espacement…).

Bon exemple : Les instructions sont renseignées sous forme de texte associé au premier élément du formulaire.

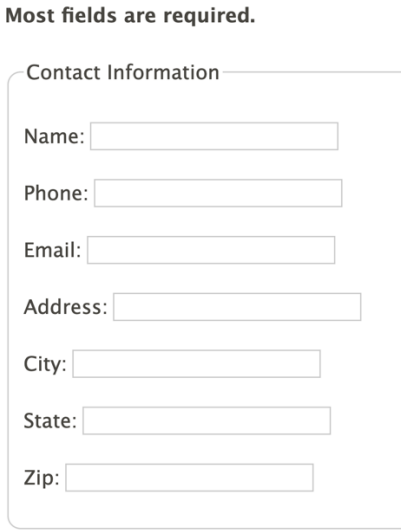


Un vrai texte est renseigné et lié au premier champ grâce à la technique décrite plus haut grâce à aria-describedby

Les instructions pour les groupes ou sections doivent être significatives.

Les instructions doivent être claires, précises et informatives. Les utilisateurs doivent connaître à l'avance les instructions nécessaires pour entrer leurs données.

Mauvais exemple : Une instruction trop vague et pas assez explicite



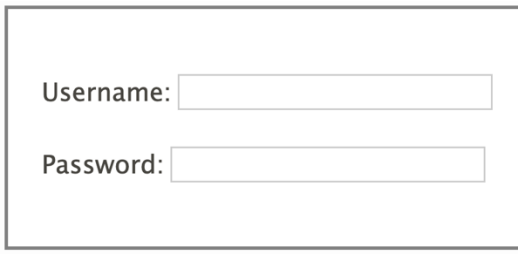
Bien que l'instruction soit correctement associée aux champs du formulaire, le texte est trop confus pour informer correctement l'utilisateur.

Les instructions pour les groupes ou sections doivent être visibles.

Les instructions doivent être visibles pour tous les utilisateurs. Les attributs aria ne s'adressent qu'aux lecteurs d'écrans et les textes figés dans une taille trop petite ou une couleur non suffisamment contrastée risquent de ne pas être perceptibles par certains internautes.

Mauvais exemple : Des instructions dans un texte caché

Dans cet exemple, les instructions sont cachées visuellement en CSS. Les utilisateurs d'un lecteur d'écran entendent les instructions qui ne sont toutefois pas visibles par les internautes voyants.

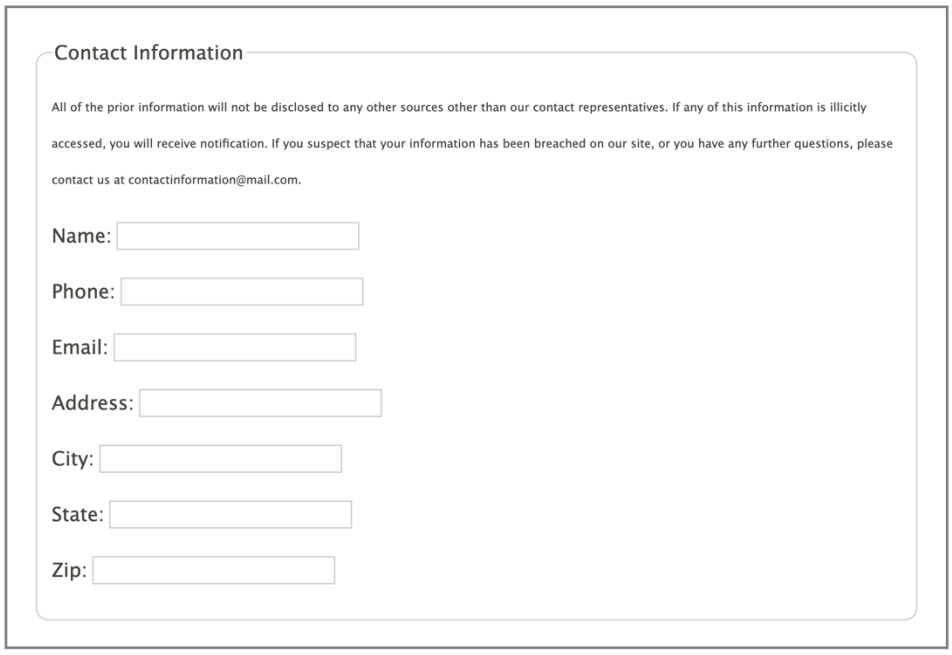



```
<div class="visually-hidden" id="desc">Must not contain spaces</div>

<p>

<label for="username">Username:</label>

<input type="text" id="username" name="username"  aria-describedby="desc">

</p>

<p>

<label for="password">Password:</label>
```

```
<input type="password" id="password" name="password" aria-describedby="desc">
</p>
```

Mauvais exemple : Texte d'instruction trop petit

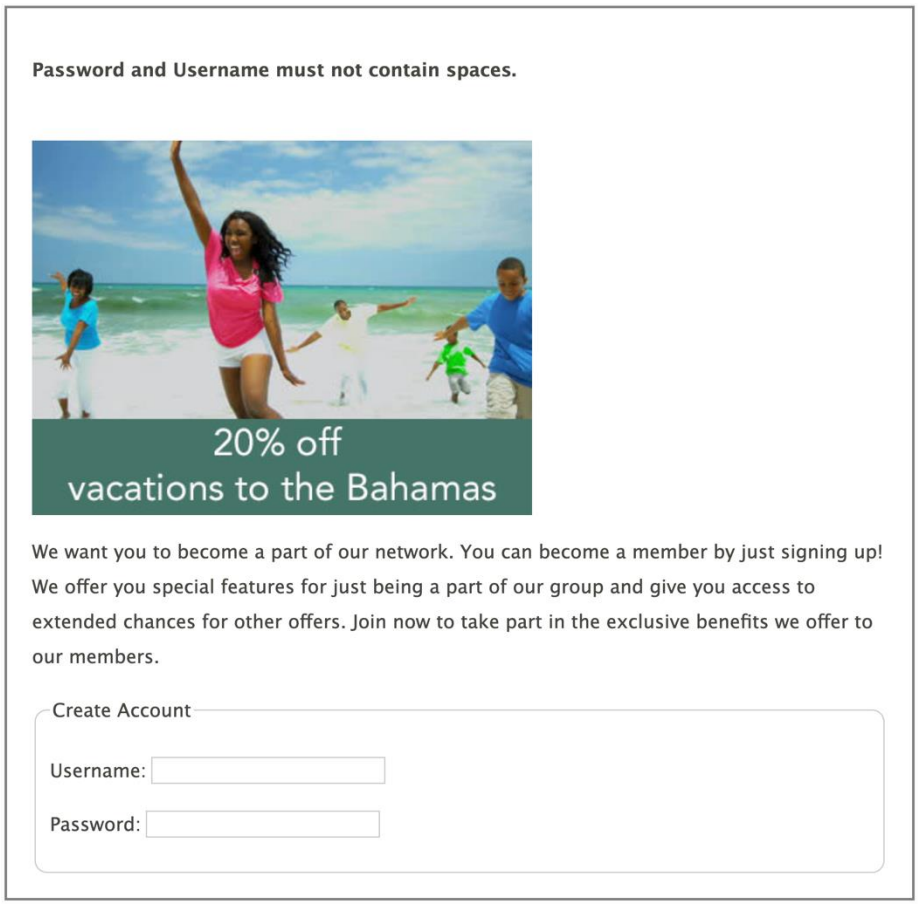
Les personnes qui n'utilisent pas de zoom risquent de ne pas pouvoir lire ce texte trop petit.



Les instructions pour les groupes ou sections devraient être visuellement proches des éléments auxquels ils font référence.

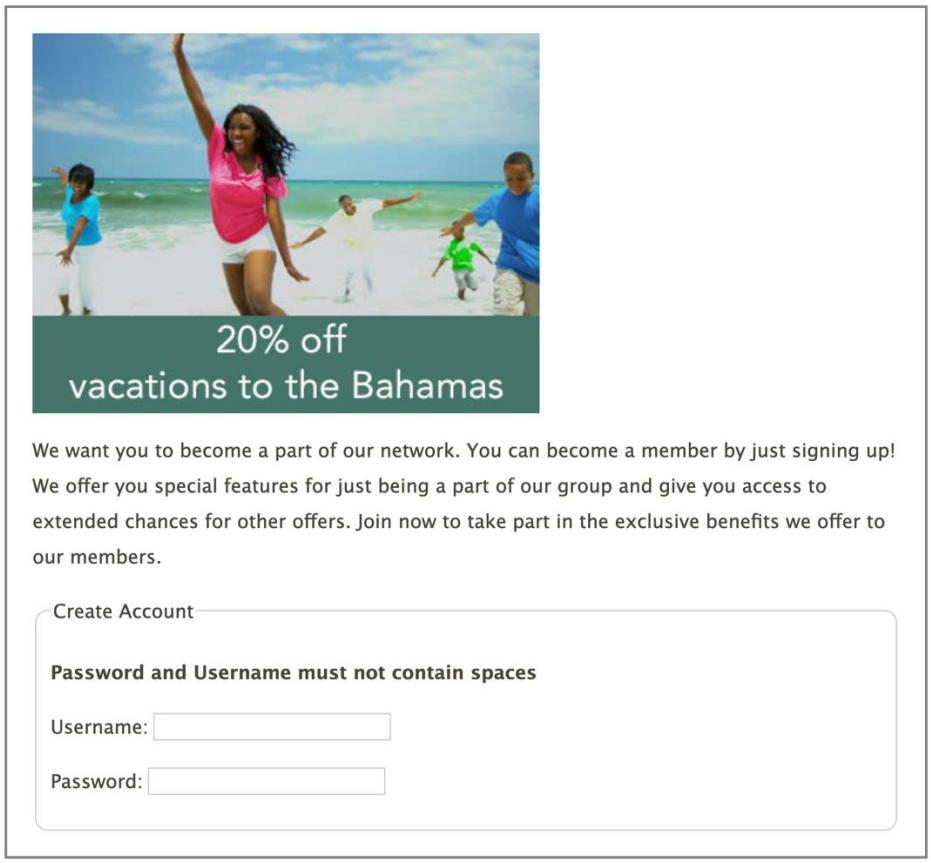
Mauvais exemple : Les instructions sont séparées du formulaire

Dans cet exemple, les instructions sont séparées par une image et un texte. Le fait que le mot de passe et le login ne peuvent pas contenir d'espace risque d'être oublié on non perçu car le message n'est plus dans le champ de vision. Sont visés, les utilisateurs de zoom, de lecteur d'écran, les personnes à la mémoire courte déficiente,… tous les utilisateurs dont l'écran n'affiche pas la totalité du formulaire.



Bon exemple : Les instructions sont renseignées juste avant les champs à compléter

Les instructions sont visuellement proches des champs concernés auxquels ils sont associés grâce à l'attribut aria-describedby.




```
<img src="/assets/images/tips/alt-family-beach.jpg"
  alt="Enjoy a 20% discount on a sun-filled, family-friendly vacation to the Bahamas">
<p>We want you to become a part of our network. You can become a member by
  just signing up! We offer you special features for just being a part of our
  group and give you access to extended chances for other offers. Join now to
  take part in the exclusive benefits we offer to our members.</p>
<fieldset>
<legend>Create Account</legend>
  <p><strong>Username and password
  <span id="mustnot">must not contain spaces</span></strong></p>
  <p><label for="username">Username:</label>
    <input type="text" id="username" aria-describedby="mustnot"></p>
  <p><label for="password">Password:</label>
    <input type="password" id="password" aria-describedby="mustnot"></p>
</fieldset>
```

Les instructions pour les groupes ou sections devraient être proches des éléments auxquels ils font référence dans le DOM.

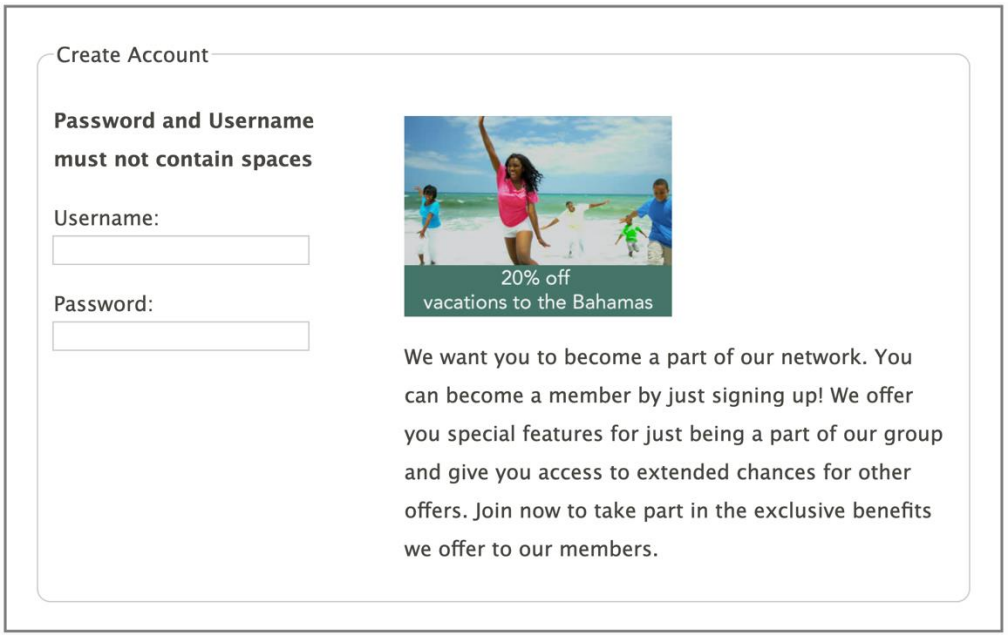
Les lecteurs d'écran lisent en général le contenu de la page dans l'ordre d'apparition dans le DOM. Les messages d'instruction doivent donc être proches des éléments auxquels ils se rapportent.

Dans la plupart des cas, si les informations sont visuellement proches de leurs éléments, la proximité est effective également dans le DOM. Mais certains positionnements en CSS peuvent montrer une proximité qui n'est pas réelle.

Mauvais exemple : Une séparation dans le DOM causée par un positionnement float en CSS

Dans cet exemple, les instructions « Username and password must not contain spaces" semblent visuellement proches de leurs champs correspondant mais ce n'est pas le cas dans le DOM.

Un paragraphe entier et une image ont été insérés dans le formulaire entre les instructions et les champs. Leur positionnement en CSS les montrent sur la droite du formulaire alors qu'ils apparaissent dans le DOM juste après le texte d'instructions.




```
<fieldset>
<legend>Create Account</legend>
  <p><strong>Username and password<br>
  must not contain spaces</strong></p>
<div id="float">
  <img src="/assets/images/tips/alt-family-beach.jpg" alt="Enjoy a 20% discount on a sun-filled, family-friendly vacation to the Bahamas">
  <p>We want you to become a part of our network. You can become a member by just signing up! We offer you special features for just being a part of our group and give you access to extended chances for other offers.
```

```
  Join now to take part in the exclusive benefits we offer to our members.
</p>
</div>
  <p><label for="username">Username:</label><br>
    <input type="text" id="username"></p>
  <p><label for="password">Password:</label><br>
    <input type="password" id="password"></p>
</fieldset>
```

b) *Instructions pour les champs de formulaires*

Les instructions pour un élément doivent être techniquement associées à l'élément

Les instructions spécifiques nécessaires pour entrer une donnée doivent être techniquement associées aux champs, boutons et contrôles. L'association peut se faire sémantiquement par le <label> associé mais aussi grâce à l'attribut aria-describedby.

Bon exemple : Les instructions sont disponibles via aria-describedby

Email: (Must be a valid email address)

```
<p>
  <label for="email">Email:</label>
  <input type="email" name="email" id="email" aria-describedby="input-instructions">
  <span id="input-instructions">(Must be a valid email address)</span>
</p>
```

Mauvais exemple : Les instructions ne sont pas associées au <input>

Email: (Must be a valid email address)

```
<p>
  <label for="email">Email:</label>
  <input type="email" name="email" id="email">
  <span>(Must be a valid email address)</p>
</p>
```

Les instructions pour un élément doivent être techniquement présentes en tant que textes explicites

Sans texte explicite associé au champ, le lecteur d'écran ne pourra pas fournir d'information à l'utilisateur. Les méthodes possibles sont l'insertion du texte dans le <label> correctement associé ou l'utilisation des attributs aria.

Mauvais exemple : Une icon font sans texte associé

Le format de fichier (Microsoft Word) est uniquement spécifié par une icône qui montre le logo Word. Sans texte alternatif, le lecteur d'écran ne peut renseigner le format souhaité.


```
<p>
  <label for="upload">Upload file</label>
  <span class="far fa-file-word fa-lg" id="desc"></span>:
  <input type="file" id="upload" aria-describedby="desc">
</p>
```

Bon exemple : une icon font avec une alternative textuelle associée

Dans le même exemple que le précédent, le texte « in MS Word format" est ajouté après l'icône pour permettre à tous les utilisateurs de comprendre le format de fichier souhaité. Ce texte est associé au champ via l'attribut aria-describedby pour permettre aux lecteurs d'écran de faire le lien entre le champ et le message.


```
<p>
  <label for="upload">Upload file</label>
  <span id="desc">in MS Word format</span> <span class="far fa-file-word fa-lg"></span>:
  <input type="file" id="upload" aria-describedby="desc">
</p>
```

Les instructions pour un élément doivent être explicites et sans équivoque

Si un champ demande un format spécifique comme par exemple un format de date, l'information doit être correctement décrite.

Ainsi, le texte « Entrer un format de date valide » n'a aucun sens. Il faudra renseigner, par exemple, « MM/YYYY ».

Remarque particulière par rapport aux formats de dates. MM et YYYY sont en anglais et il faudra les traduire dans toutes les langues. Il est par ailleurs recommandé de donner un exemple réel comme « 11/2022 »

Les instructions pour un élément doivent être visibles, suffisamment grandes, proches de leur élément correspondant visuellement et dans le DOM, ne pas donner une indication selon par un élément sensoriel (forme ou couleur)…

Ces critères ont été décrits et illustrés précédemment.

c) Champs requis

Les champs obligatoires clairement identifiés permettent à l'utilisateur de gagner du temps au moment du remplissage du formulaire et d'éviter les allers-retours et risques d'erreurs.

Généralement, le caractère obligatoire sera renseigné sur le champ <input> grâce à l'attribut aria-required= « true » doublé d'une identification visuelle pour les personnes qui n'utilisent pas de lecteur d'écran.

Les champs obligatoires devraient être techniquement programmés comme tels

Aria-required= « true »

Cet attribut utilisé sur un champ est un très bon moyen pour renseigner son caractère obligatoire. Le lecteur d'écran dira « obligatoire » ou « requis » après avoir lu le label du champ. Cette donnée est par contre invisible et devra être doublée par un indice visuel.

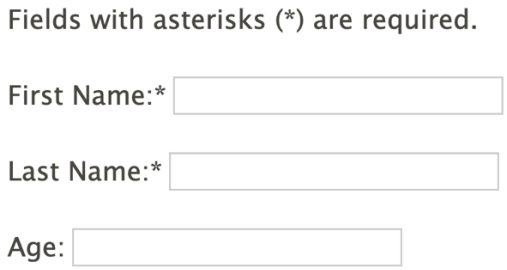
L'attribut html « required »

Le comportement pour les utilisateurs de lecteurs d'écran est le même que pour aria-required= « true » mais ajoute un comportement visuel supplémentaire qui force l'utilisateur à compléter le champ si celui-ci est resté vide.

Tous les navigateurs ne supportent par contre pas cet attribut et, comme pour aria-required, le caractère obligatoire doit quand même être renseigné au préalable pour les utilisateurs voyants.

Aria-required semble donc plus indiqué sachant qu'il est demandé de ne pas utiliser required et aria-required= « true » sur le même champ pour éviter la redondance de l'information pour les utilisateurs de lecteurs d'écrans.

Bon exemple : Les champs requis utilisent aria-required= « true » et une astérisque visuelle




```
<p>Fields with asterisks (*) are required.</p>
<p>
<label for="firstname">First Name:*</label>
<input id="firstname" name="firstname" type="text" aria-required="true">
</p>
<p>
<label for="lastname">Last Name:*</label>
<input id="lastname" name="lastname" type="text" aria-required="true">
</p>
<p>
<label for="age">Age:</label>
<input id="age" name="age" type="text">
</p>
```

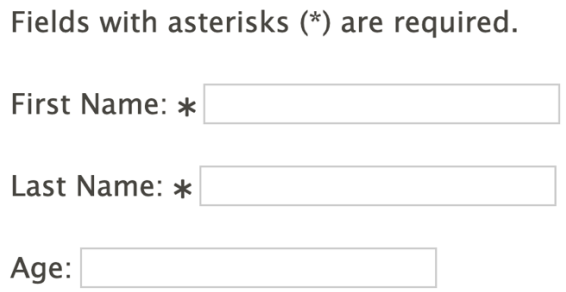
Remarques :

- Le fait que les champs obligatoires sont identifiés par une astérisque doit être renseigné avant remplissage du formulaire.
- Le lecteur d'écran lira l'astérisque du <label> et le mot « obligatoire » ou « requis » lié à aria-required= « true ». Ce comportement n'est pas bloquant mais le caractère redondant de l'information peut être contré par la technique décrite dans l'exemple suivant.
- L'astérisque (*) peut être remplacée par le mot « (obligatoire) » à renseigner en toutes lettres dans le label des champs concernés.

Bon exemple : Les champs requis utilisent aria-required= « true » et une indication visuelle invisible pour les lecteurs d'écrans.

Cette technique est sensiblement la même que dans l'exemple précédent mais, dans ce cas, l'utilisateur n'entendra pas l'astérisque présent dans les labels puisqu'elle est représentée par une image dans une classe CSS.

Une interrogation pourrait toutefois surprendre l'utilisateur qui pourrait entendre en début de formulaire que les champs avec astérisque sont obligatoires puis ne pas entendre les astérisques par la suite. Le <p> qui contient « Fields with asterisks (*) are required » pourrait toutefois être cache aux lecteurs d'écrans grâce à l'attribut aria-hidden= « true » pour rendre la solution complète.




```
<p>Fields with asterisks (*) are required.</p>
<p>
<label for="firstname">First Name: <span class="fa fa-asterisk"></span></label>
<input id="firstname" name="firstname" type="text" aria-required="true">
</p>
<p>
<label for="lastname">Last Name: <span class="fa fa-asterisk"></span></label>
<input id="lastname" name="lastname" type="text" aria-required="true">
</p>
<p>
<label for="age">Age:</label>
<input id="age" name="age" type="text">
</p>
```

Le processus de validation du formulaire doit inclure un message d'erreur qui identifie visuellement et techniquement le champ obligatoire non complété après la tentative de soumission.

Un message clair compréhensible visuellement et par les lecteurs d'écran doit identifier les champs obligatoires qui n'ont pas été complétés après l'envoi du formulaire.

Remarques :

- L'utilisateur doit avoir un feed-back sur la réussite de l'opération : réussi ou erreurs dans le formulaire (identification dans le title de la page, position du focus sur le premier message d'erreur…).
- L'utilisateur doit pouvoir identifier facilement les champs en erreur sans re-encoder toutes les informations ou repasser systématiquement dans tous les champs.
- Les messages d'erreurs devraient indiquer, dans le cas d'un format non valide, un exemple réel de donnée à encoder.
- Le message d'erreur doit être visible par tous les utilisateurs (texte, contraste suffisant, taille et position du message, utilisable par les technologies d'assistance…)

Bon exemple : Les messages d'erreur sont identifiés visuellement et techniquement

Dans cet exemple, les messages d'erreur visuels sont techniquement associés aux champs concernés via l'attribut aria-describedby.

Remarques :

- Un exemple réel de date devrait être renseigné dans le message d'erreur.
- L'attribut value= « User name » n'est pas obligatoire et peut même générer des erreurs de compréhension chez certains utilisateurs. Laisser le champ vide est recommandé.

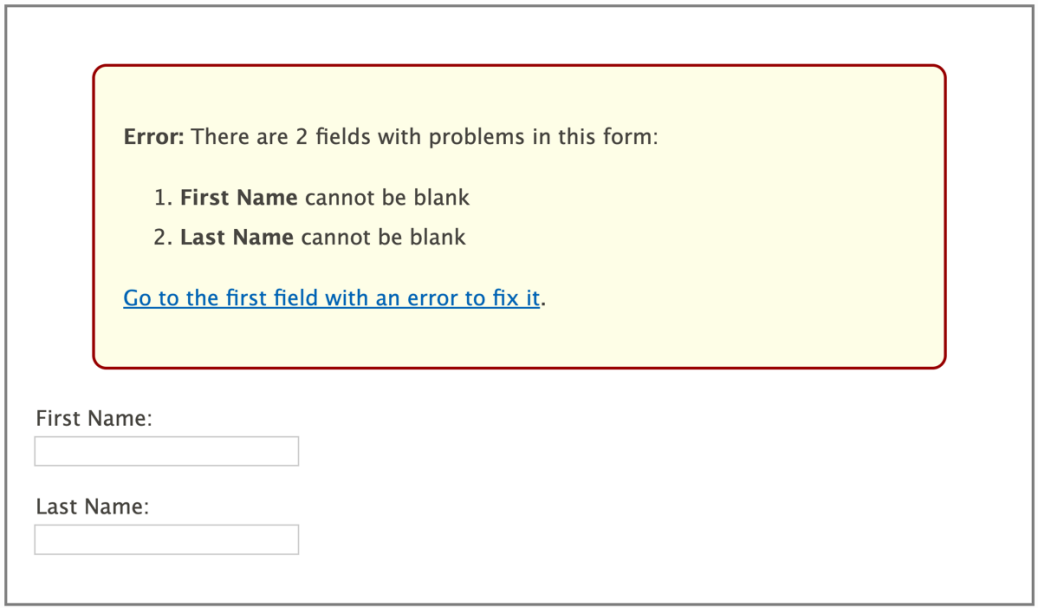

```
<p>
  <label for="username">Username:</label>
  <input type="text" name="username" id="username" value="User name">
</p>
<p class="error">
  <label for="expire">Expiration:</label>
  <input type="text" name="expire" id="expire"
    aria-required="true"
    aria-invalid="true"
    aria-describedby="expDesc">
  <span id="expDesc">Error: Date is required.</span>
```

</p>

Bon exemple : Un résumé des erreurs

Cet exemple montre un résumé des erreurs après avoir tenté d'envoyer le formulaire. Ses caractéristiques sont les suivantes :

- Information présente sur le nombre d'erreurs présentes
- Explication claire sur le champ en erreur
- Explication claire sur le type d'erreur
- Lien explicite vers le premier champ qui contient une erreur
- Utilisation d'aria-required pour identifier les champs obligatoires
- Utilisation d'aria-invalid pour identifier qu'un champ est en erreur (vide ou format non conforme). En combinaison avec aria-required, le paramètre ne doit pas être positionné en « true » tant que le formulaire n'a pas été envoyé. La paramètre « true », « false », « grammar » ou « spelling » sera modifié en Javascript.




```
<div class="alert">
<p><strong>Error:</strong>
 There are 2 fields with problems in this form:</p>
 <ol>
    <li><strong>First Name</strong> cannot be blank</li>
    <li><strong>Last Name</strong> cannot be blank</li>
 </ol>
<a href="#fname" id="errorSummaryLink">Go to the first field with an error to fix it</a>.
</div>
<label for="fname">First Name:</label><br>
<input type="text" id="fname" aria-required="true" aria-invalid="true">
<label for="lname">Last Name:</label><br>
```

<input type="text" id="lname" aria-required="true" aria-invalid="true">

L'objet des champs communément utilisés qui collectent des informations personnelles doit être techniquement défini sur base des ressources officielles du W3C en matière de composants d'interfaces.

L'objectif de ce critère est d'aider l'utilisateur à compléter les champs de formulaires plus rapidement et sans erreurs en auto-complétant ses données personnelles correctement identifiées dans les bons champs.

L'implémentation de ce critère se fait par l'utilisation, dans l'<input> correspondant, de l'attribut autocomplete= «paramètre ».

La liste des paramètres utilisables est publiée sur le site du W3C

Exemple d'utilisation d'autocomplete :

<input class="loginPassword" type="password" id="loginPassword" name="password" aria-required="true" autocomplete="current-password" required="required" oninvalid="this.setCustomValidity('Enter password to login')" oninput="this.setCustomValidity('')"/>

Formulaires dynamiques

a) *Formulaires progressifs*

Dans certains cas, des informations sont demandées aux utilisateurs en plusieurs étapes et les réponses de chaque étape impactent les autres étapes du formulaire.

Pour rendre ce processus accessible, les bonnes pratiques suivantes doivent être prises en considération :

Changer les options futures mais pas les options déjà passées.

Les utilisateurs de lecteurs d'écran lisent le contenu de manière plus linéaire. Lorsqu'ils sont passés à un endroit du formulaire, ils ne reviendront pas en arrière pour vérifier qu'un changement a eu lieu.

Il faut donc s'assurer que les changements d'options n'impacteront que ce qui se trouve dans le DOM à partir de la position de l'utilisateur.

Autoriser l'utilisateur à modifier les choix effectués dans les étapes précédentes.

Dans certains cas, l'utilisateur devra pouvoir modifier les réponses données dans les étapes précédentes. Le formulaire doit lui permettre de naviguer en arrière et de corriger ses réponses.

Limiter le nombre d'étapes par écran

Trop d'étapes ou de questions sur une seule page peuvent être perturbantes et générer des problèmes de navigation ou de compréhension.

Les questions devraient être rassemblées par étape cohérente et une navigation doit permettre de naviguer facilement entre les étapes pour revoir les questions / réponses. Une autre solution pratique consiste à montrer une synthèse des questions / réponses (sous forme de navigation) et de permettre à l'utilisateur de revenir dans chaque question en cliquant directement sur le lien correspondant.

Renseigner le nombre d'étapes

Les utilisateurs apprécient de connaître à tout instant le numéro d'étape où ils sont arrivés et le nombre total d'étapes. Ce système doit bien entendu être accessible également par les lecteurs d'écran.

Permettre la navigation au clavier

Gérer correctement le focus à chaque étape du processus. En fonction de la manière dont le formulaire est construit, le focus devra être placé sur l'élément le plus logique possible pour permettre une navigation fluide et accessible. Si l'élément n'est pas focusable nativement, par exemple un niveau de titre (heading) ou un container, le focus peut être implémenté via tabindex= « -1 ».

Les boutons qui permettent de naviguer d'étape en étape doivent pouvoir recevoir le focus également.

Veiller aussi à rendre le focus visible (règles de contraste, bordure…). Les propriétés CSS de survol doivent également être implémentées sur la position focus.

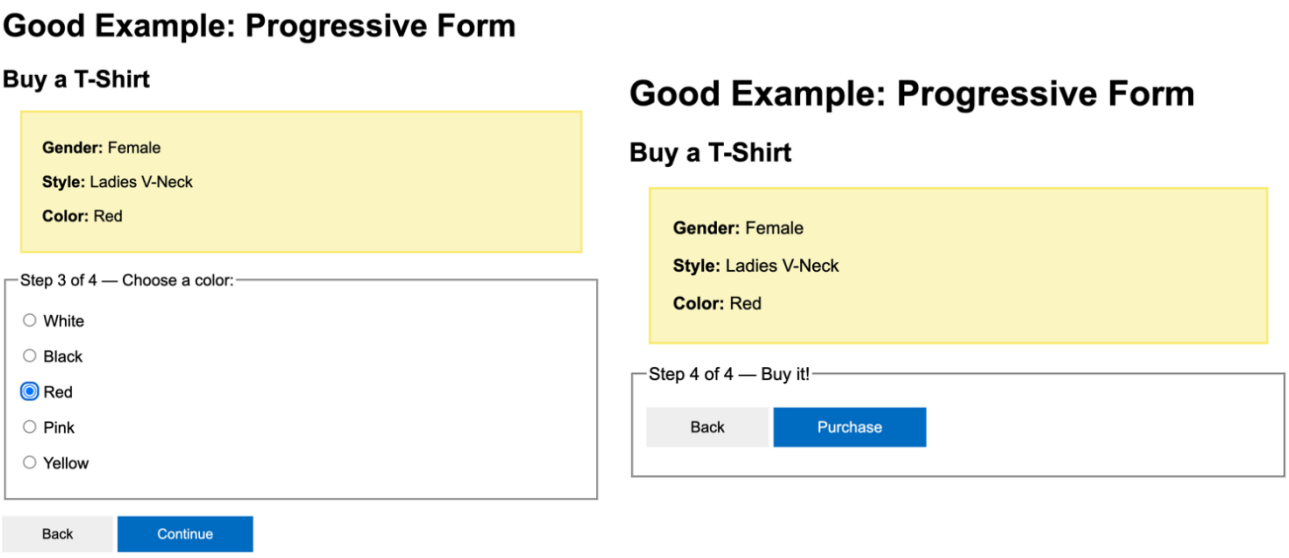
Envisager la mise en place d'une synthèse concise des réponses

Il peut être fastidieux de passer en revue toutes les étapes d'un formulaire pour se souvenir des données encodées avant l'envoi. Une synthèse concise facilite ce processus et rassure l'utilisateur tout en lui faisant gagner du temps.

Il peut être également intéressant d'associer techniquement la synthèse des réponses au bouton d'envoi comme dans l'exemple de code ci-dessous :

Summary goes here ...

<button aria-describedby="summary">Submit</button>



b) Changements de contexte

Les utilisateurs peuvent être désorientés par un changement de contexte inattendu. Un changement de contexte peut être défini par un changement sur :

- Le user agent : Par exemple lorsque le clic sur un lien ou un bouton fait quitter le navigateur et ouvre une autre application.
- Le viewport : Par exemple lorsque l'écran monte ou descend, lorsqu'un élément remplace visuellement un autre élément ou change de position ou lorsqu'une nouvelle fenêtre s'ouvre.
- Le focus : Par exemple lorsque le focus se place sur un autre élément.
- Le contenu : Par exemple lorsque la signification d'une page change significativement, ou quand les éléments de la page sont réorganisés.

Tous ces changements sont autorisés mais, en accessibilité numérique, ils ne peuvent s'opérer automatiquement sans que l'utilisateur ne soit conscient qu'un changement va se produire et sans qu'il ne connaisse la nature du changement et l'impact de son action.

Le focus ou le survol sur un élément ne peut pas générer automatiquement un changement de contexte, sauf si l'utilisateur a été prévenu à l'avance de la nature du changement.
Lier un comportement inattendu à un focus sur un élément peut désorienter tous les internautes et en particulier les utilisateurs de lecteurs d'écran.

Mauvais exemples :

- Le focus sur un champ de formulaire ouvre automatiquement une fenêtre qui donne des informations sur la manière de compléter le champ.
- Le hover sur une zone de la page déclenche automatiquement une fenêtre.

Un changement de valeur dans un élément ne peut pas générer automatiquement un changement de contexte, sauf si l'utilisateur a été prévenu à l'avance de la nature du changement.
Note : Le changement de contexte est autorisé lors du clic sur un bouton d'envoi de formulaire (ou équivalent) car l'utilisateur a fini son action et s'attend à ce changement.

Mauvais exemples :

- Une liste <select> est utilisée comme moyen de naviguer sur le site. Le contenu de la page change dès que l'utilisateur sélectionne une des options.
- Un formulaire contient 3 champs pour renseigner un numéro. Lorsque le nombre de chiffres maximum est atteint dans un champ, le focus saute automatiquement dans le champ suivant sans prévenir l'utilisateur.

c) Champs de formulaires personnalisés

Les ressources ARIA couvrent en détail les techniques pour implémenter les différents widgets en Javascript. Nous listons ici les points d'attention principaux pour implémenter ces composants.

Les éléments natifs HTML devraient être utilisés en priorités lorsque c'est possible.
Les éléments de formulaire HTML sont accessibles par nature sans ajout de JavaScript pour les rendre utilisables au clavier ou par les technologies d'assistance.

Par ailleurs, les utilisateurs ont l'habitude de les utiliser et connaissent leurs comportements par défaut sans obligation de fournir d'instructions spécifiques, sans courbe d'apprentissage.

Les éléments de formulaires personnalisés devraient fonctionner et réagir le plus possible comme les éléments natifs HTML.
Par exemple, une case à cocher (checkbox) doit pouvoir être cochée / décochée via la barre d'espace et le message doit être proche de celui annoncé par les lecteurs d'écran. Attention surtout à rendre chaque composant utilisable au clavier.

Les comportements attendus de chaque composant sont disponibles sur le site du W3C-WAI

Les éléments de formulaires personnalisés devraient avoir un nom accessible, un rôle et une valeur.
L'ensemble de ces informations est disponible en ligne via les ressources ARIA specifications

Nom (Name) :
Le label de l'élément qui renseigne son nom accessible comme par exemple « date picker », « Submit »… est souvent repris dans un aria-label ou aria-labelledby.

Rôle (Role) :
Le but ou fonction de l'élément comme par exemple « dialog », « menu » pu « tabpanel »… doit être défini. La liste des rôles est disponible sur le site W3C-ARIA

Valeur (Value) :
Les attributs ou l'état courant de l'élément comme par exemple aria-selected=« true » ou aria-expanded=« false ». Les valeurs doivent être mises à jour lorsque c'est nécessaire (true devient false…)

Les mises à jour et changements d'état qui ne peuvent être communiqués par les méthodes HTML ou ARIA devraient être communiquées via des messages ARIA live.
Les lecteurs d'écran doivent connaître les changements apportés à la page. Par exemple : « tableau trié par nom, ascendant » ou « Résultats triés par popularité »… Ces changements peuvent alors être restitués aux utilisateurs.

Les différentes valeurs et attributs aria-live sont disponibles sur le site du W3C-ARIA

Validation des formulaires

Malgré les meilleures instructions possibles, nous ne pourrons pas éviter les erreurs dans les formulaires. Un aspect important en accessibilité est le traitement des erreurs. Les WCAG 2.1 insistent beaucoup sur la manière d'alerter les utilisateurs pour les aider à corriger leurs erreurs. Certains points importants sont à considérer :

- Donner un feedback positif ou négatif à tous les utilisateurs.
- Conserver un maximum possible des informations correctes
- Afficher les messages d'erreur dans leur contexte pour une correction rapide.
- Donner aux utilisateurs suffisamment d'informations pour corriger leurs erreurs. Les messages doivent :
  - Être clairs
  - Attirer l'attention
  - Être stylés distinctivement des champs d'input
  - Être complètement accessibles

a) Prévention des erreurs : Bonnes pratiques

L'objectif est de construire des formulaires suffisamment explicites pour éviter les erreurs des utilisateurs. Ce guide a déjà couvert beaucoup de techniques de prévention. Voici une synthèse :

- Le texte des labels sont précis et disponibles pour tous les utilisateurs
- Si le texte du label ne donne pas suffisamment d'instructions sur l'objet d'un champ, donner des instructions complémentaires.
- Lorsque c'est nécessaire et approprié, restreindre le type de données attendues dans le champ : exemples : type texte pour un code postal, listes déroulantes, boutons radio ou cases à cocher.
- Proposer des vues du formulaire étape après étape pour ne pas noyer l'utilisateur par trop de données en même temps.
- Utiliser des design patterns conventionnels que l'utilisateur connaît et pourra prendre en main sans trop de temps d'apprentissage. Utiliser des éléments HTML natifs le plus possible.
- Aider l'utilisateur à concentrer son attention en positionnant les labels et instructions proches de leurs champs respectifs, en réduisant les distractions et en utilisant des éléments pour guider le guider dans son processus. Rendre les messages interactifs visuellement évidents.
- Synthétiser les informations qui seront communiquées avant le bouton d'envoi et proposer à l'utilisateur de vérifier ses données, surtout pour des actions irréversibles.
- Afficher à l'écran une synthèse du résultat après l'envoi.
- Si possible, permettre à l'utilisateur de restaurer les données dans leur état initial soit par une solution technique « undo », soit en contactant un intervenant soit par un procédé qui permet de corriger ses données.
- Si c'est approprié, envoyer un email de confirmation à l'utilisateur comme moyen de contrôle supplémentaire et pour le rassurer sur le suivi de son traitement.

b) Identification des erreurs

Les feedbacks d'erreurs devraient être disponibles directement après l'envoi du formulaire.

L'utilisateur doit connaître les éventuelles erreurs directement après l'envoi. Sans retour sur ces informations, l'utilisateur pourrait croire que le formulaire ne fonctionne pas et abandonner son action. C'est principalement le cas pour les utilisateurs de lecteurs d'écran.

Les erreurs peuvent être communiquées après l'envoi complet, étape par étape ou lors de chaque remplissage de champs.

Les feedbacks d'erreurs devraient être techniquement associés à l'élément concerné.

Si les messages d'erreurs sont associés à leur élément correspondant, l'utilisateur les retrouvera plus facilement pour une correction plus rapide.

Bon exemple : Le message d'erreur est associé techniquement à son élément

Dans cet exemple, l'attribut aria-describedby associe le message d'erreur au champ concerné


```
<div class="formborder">

<div class="error">
```

```
  <span class="fa fa-exclamation-triangle" role="img" aria-label="Error: "></span>
  <span id="errorMsg">Please Enter a Name for This Contact Person</span>
</div>
<p>
  <label for="name">Name: </label><br>
  <input type="text" name="name" id="name" aria-invalid="true" aria-describedby="errorMsg">
</p>
</div>
```

Les feedbacks d'erreurs doivent être techniquement déterminables.

Le texte du message techniquement associé doit être suffisamment clair et explicite. Les messages peuvent, selon les cas, être renseignés dans des attributs alt, aria-label, aria-labelledby et aria-describedby.

Sinon, les utilisateurs de lecteurs d'écran ne pourront pas corriger leurs erreurs.

Mauvais exemple : Un message d'erreur sans message techniquement déterminable

Dans cet exemple, même si une icône de triangle avec point d'exclamation signale visuellement une erreur, il n'est pas possible pour l'utilisateur d'un lecteur d'écran de comprendre le message d'erreur.


```
<span class="fa fa-exclamation-triangle"></span><label for="name"> Name: </label><br>
<input type="text" name="name" id="name" aria-invalid="true">
```

Bon exemple : Un message d'erreur avec un message techniquement déterminable

Dans cet exemple, une icône de triangle avec point d'exclamation signale visuellement une erreur et aria-describedby est utilisé pour fournir une information sur la nature de l'erreur disponible pour tous.


```
<span class="fa fa-exclamation-triangle"></span>
<label for="name">Name:</label><br>
<input type="text" name="name" id="name"
  aria-invalid="true" aria-describedby="errorMsg"><br>
<span id="errorMsg" class="errorMsg">Error: Please enter a name"</span>
```

Les feedbacks d'erreurs doivent être explicites et significatifs.
Les messages d'erreur doivent être clairs, appropriés et fournir suffisamment d'informations sur l'action à réaliser pour corriger l'information.

Exemple de bon message : « L'adresse email ne peut pas être vide »

Exemple de mauvais message : « Impossible de créer l'utilisateur suite à une erreur »

Les feedbacks d'erreurs doivent être visibles.
Les messages doivent être techniquement associés à leur champ et être utilisables par les lecteurs d'écran mais ils doivent également être visibles par tous les utilisateurs.

Mauvais exemple : Un message d'erreur disponible uniquement pour les lecteurs d'écran

Cet exemple montre un champ entouré d'une bordure rouge pour signaler une erreur. Outre le fait que tous les utilisateurs ne peuvent discerner la couleur, la nature de l'erreur n'est visible que par les lecteurs d'écran via l'attribut aria-label.


```
<label for="date">Date: </label>

<input type="text" name="date" id="date" aria-invalid="true"

  aria-label="Error: Please enter the date as MM/YYYY">
```

c) Confirmation de réussite de soumission de formulaire

Les messages de confirmation d'envoi de formulaire devraient être techniquement déterminables.
Comme pour les autres messages vus dans les autres chapitres, le message de confirmation doit être techniquement déterminable pour pouvoir être compris par les lecteurs d'écrans.

Bon exemple : Un message de succès d'envoi techniquement déterminable

L'exemple montre une icône verte de validation associée à un aria-label qui renseigne le statut de l'envoi.

Cette icône est doublée d'un texte dans un paragraphe visible par tous.


```
<div class="success">

<p><span class="fa fa-check fa-4x" aria-label="Success"></span></p>

<p>Your form has been successfully submitted.</p>

</div>
```

Les messages de confirmation d'envoi de formulaire devraient être explicites et significatifs.
Les messages de confirmation d'envoi doivent être clairs, appropriés et informatifs pour confirmer à l'utilisateur le bon suivi de sa demande.

Les informations à communiquer sont, par exemple :

- Le formulaire est-il bien envoyé ?
- Les données ont-elles été effacées ?
- Les préférences ont-elles été sauvées ?…

Les messages de confirmation d'envoi de formulaire doivent être visibles.
Tous les utilisateurs, voyants ou non, doivent pouvoir accès aux messages de confirmation d'envoi.

Mauvais exemple : Un message de succès d'envoi via aria-live dans un conteneur caché

Une région aria-live peut être utilisée pour annoncer en temps réel aux lecteurs d'écran que les préférences ont été sauvegardées. L'utilisateur entendra cette information.

Dans cet exemple, la région aria-live est malheureusement incluse dans un conteneur caché via une classe « visually-hidden ». De ce fait, les personnes voyantes ne pourront pas accéder au message de confirmation.

```
<div id="aria-live" aria-live="assertive" class="visually-hidden">
    Your preferences have been saved.</div>
```

Sémantique et structure

Le <Title> de la page

Le title de la page est très important et critique. Il contient la première information lue par le lecteur d'écran, il permet d'identifier visuellement la page lorsque plusieurs onglets sont ouverts dans le navigateur et il permet de retrouver facilement une page enregistrée ou consultée dans les favoris et l'historique du navigateur.

Outre les besoins spécifiques des personnes aveugles, dont la mémoire est défaillante ou qui ont des problèmes d'attention, un title correctement utilisé est bénéfique à tous et permet d'améliorer le référencement de la page.

a) Un <title> pour toutes les pages

Le <title> de la page doit être présent et contenir du texte
Bon exemple : un <title> de page valide

```
<!DOCTYPE html>

<html lang="en">

<head>

<title>Products and Services | My website</title>

</head>
```

Mauvais exemples : title manquant ou vide

```
<!DOCTYPE html>

<html lang="en">

<head>

</head>
```

```
<!DOCTYPE html>

<html lang="en">

<head>

<title></title>

</head>
```

Le <title> de la page doit être mis à jour à chaque changement d'adresse.

Le changement de l'adresse de la page du au chargement d'une nouvelle page, à un appel en AJAX ou à un événement JavaScript doit entrainer la mise à jour du <title> de la page pour le rendre cohérent avec le contenu affiché.

b) Un <title> de page significatif

Le <title> de la page doit être précis et informatif

Le title doit identifier clairement le but et le contenu de la page pour éviter aux utilisateurs de se tromper de page ou de chercher une information qui ne se trouvera pas dans le contenu de la page.

Des contenus du type « Untitled Document » ou « Document Content » ne sont donc pas permis.

Si la page est le résultat d'une action de l'utilisateur ou d'un changement de contexte programmé, le contenu du <title> devrait définir le résultat du changement de contexte à l'utilisateur.

Bon exemple : Un title qui contient le mot recherché dans un moteur de recherche

```
<title>"Warranty" - Search Results</title>
```

Mauvais exemple : Un title qui ne contient pas le mot recherché dans un moteur de recherche

```
<title>Search Results</title>
```

Le <title> devrait être concis

La totalité du contenu du title sera lu par le lecteur d'écran, visible dans l'onglet du navigateur, les favoris et l'historique. La balise ne contient pas de limite maximum mais il est recommandé d'être le plus concis possible tout en donnant le sens de la. 80 caractères semble être un maximum acceptable.

Bon exemple : Un title concis

```
<title>Accessibility Services | Accessia</title>
```

Mauvais exemple : Un abus de mots clés plutôt qu'un bon title

```
<title>Mortgages Inc. - Mortgage services, best mortgages in New York, low rates, low API, easy financing, variable rates, fixed rate mortgages</title>
```

Le <title> devrait être unique et commencer par l'information principale de la page et unique sur le site

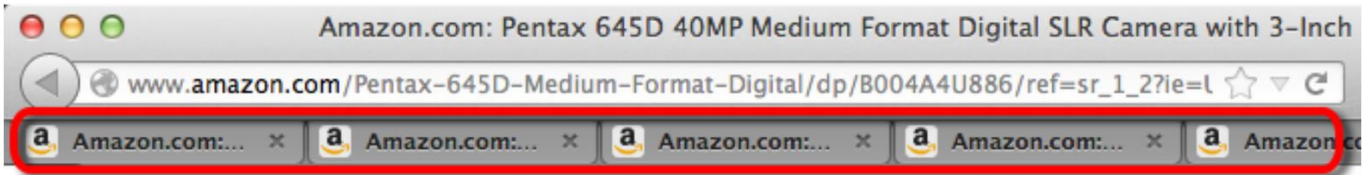
Pour éviter les confusions, accélérer la lecture de la page, la retrouver plus facilement dans les navigateurs et améliorer le référencement, le title de la page devrait être unique (pas deux title identiques par site) et commencer par l'information principale de la page et unique sur le site.

Une bonne pratique consiste à commencer par le titre principal de la page <h1> (ou par un contenu très similaire) présent dans le <main> de la page et de terminer par le nom du site. Un séparateur peut séparer les deux parties pour plus de visibilité.

On privilégiera donc plutôt un title du type : <title>Accessibility Services | Accessia</title> plutôt que <title>Accessia | Accessibility Services</title>

Mauvais exemple : Les titles des pages du site Amazon.com commencent par le nom du site

Sur le site d'Amazon.com, le title des pages commence par Amazon.com et ne permet pas de voir le nom des pages ouvertes dans le navigateur. Le lecteur d'écran commencera dira d'abord « Amazon.com… » avant le nom de la page.



La langue de la page

La plupart des lecteurs d'écran peuvent lire plusieurs langues. Lors de l'installation, l'utilisateur choisit une langue par défaut que le lecteur d'écran utilisera si la langue de la page n'est pas spécifiée.

Un mauvais choix de langue peut entrainer une incompréhension de la page à cause d'un accent non approprié ou des erreurs de compréhension de certains mots utilisés dans plusieurs langues mais qui ne signifient pas la même chose.

Un bon code langue aide par ailleurs les traducteurs à identifier la bonne langue dès le départ et participe à un meilleur référencement.

a) La langue principale de la page

La langue principale de la page doit être identifiée dans la balise <html> par une valeur valide.

Bon exemple d'implémentation de la langue de la page

```
<!doctype html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>I Have a Dream, Excerpt</title>

</head>

<body>

<p>English text…</p>
```

```
</body>

</html>
```

b) La langue d'une partie de la page

Le texte d'une page écrit dans langue différente de la langue principale doit être identifié dans un attribut lang valide.

L'attribut lang peut s'appliquer à un élément de type block (, <h1>, <p>, <table>…) ou inline (, <a>, …).

Bon exemple d'implémentation d'un changement de langue dans la page

```
<!doctype html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>I Have a Spanish Dream</title>

</head>

<body>

<p>English text…</p>

<p>While in Spain, my friend tried to speak Spanish, but she wasn't very good. Everyone kept saying

<span lang="es">No comprendo nada de lo que dices.</span></p>

</body>

</html>
```

c) Les codes langue doivent être valides

Les codes langues sont normalisés et doivent être utilisés correctement. Le code du pays (country code) peur être ajouté pour coller davantage aux réalités du pays tels que les accents particuliers, les nombres (nonante ou quatre-vingt-dix…). Tous les lecteurs d'écran ne s'adaptent toutefois pas aujourd'hui au code pays.

Bon exemple d'implémentation de la langue française de Belgique

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

Les régions (landmarks)

HTML5 permet d'identifier sémantiquement les grandes régions d'une page web comme <header>, <nav>, <main> et <footer>. ARIA permet également d'identifier des régions grâce, respectivement, aux attributs suivants : role=« banner », « navigation », « main » et « contentinfo ».

Les lecteurs d'écran permettent de naviguer de région en région. Malheureusement, les navigateurs ne proposent aujourd'hui pas d'option qui permet à tous les utilisateurs (voyants) de navigue de cette manière. Le « lien d'évitement » ou « skip links » reste donc toujours très utile pour les personnes qui ne peuvent utiliser la souris.

a) Les landmarks et / ou rôles ARIA devraient être utilisés pour designer les grandes sections d'une page web

L'utilisation des régions permet une navigation plus aisée par les lecteurs d'écran, simplifie la maintenance du design de par la standardisation du code (CSS) et participe à un meilleur référencement.

Dans le cas d'une correction de code, il peut être plus facile d'ajouter des roles ARIA sur des div existantes plutôt que d'utiliser les landmarks natifs.

L'ajout d'un rôle ARIA à un landmark natif HTML5 renforce la robustesse de l'accessibilité du code. Exemple : <header role=« banner »>

b) Les correspondances HTML5 et équivalents ARIA.

HTML5	Rôles ARIA role=«… »	Listage par les lecteurs d'écrans	Obligatoire accessibilité
<header>	banner	Oui	Oui
<nav>	navigation	Oui	Oui
n/a	search	Oui	Oui si disponible
<main>	main	Oui	Oui
<footer>	contentinfo	Oui	Oui
<aside>	complementary	Oui	Non
<section>	region	Listage pour la plupart si un nom est ajouté via aria-label ou aria-labelledby. Si le code est bien structuré à l'aide de headings (niveaux de titres <h1> à <h6>, les utilisateurs navigueront prioritairement de titre en titre. Il n'est pas recommandé de multiplier les landmarks dans la page car la lecture en est ralentie.	Non
<article>	article	Oui pour JAWS	Non
<form>	form	Listage pour la plupart des lecteurs d'écran seulement avec l'ajout du role=« form »	Oui si disponible

c) Bonnes pratiques pour les landmarks

Tous les textes devraient être contenus dans une région et cette région devrait être appropriée au contenu

Bon exemple : <header> This is the header. </header>

Mauvais exemple : This is the header.

Les instances de même type de landmark devraient être distinguées techniquement par un label (aria-label ou aria-labelledby)

Le but est de renseigner au mieux l'utilisateur de lecteur d'écran sur le contenu de la région où il se trouve.

Bon exemple : Plusieurs navigations correctement labellisées

```
<nav aria-label="Main Menu">
  <ul>
    <li><a href="index.html">Home</a></li>
    <li><a href="products.html">Products</a></li>
    <li><a href="services.html">Services</a></li>
  </ul>
</nav>
<main>
  <div>[...other content...]</div>
    <nav aria-label="Products list">
      <ul>
        <li><a href="1.html">Product 1</a></li>
        <li><a href="2.html">Product 2</a></li>
        <li><a href="3.html">Product 3</a></li>
        <li><a href="4.html">Product 4</a></li>
      </ul>
    </nav>
  </div>
  <div>[...other content...]</div>
</main>
<footer>
  <nav aria-label="Corporate and legal info">
    <ul>
        <li><a href="about.html">About us</a></li>
        <li><a href="copyright.html">Copyright notice</a></li>
        <li><a href="terms.html">Terms of use</a></li>
        <li><a href="contact.html">Contact us</a></li>
    </ul>
  </nav>
```

</footer>

Une page ne devrait contenir qu'une seule instance de type banner, main et contentinfo
La spécification ARIA précise que les rôles : role=« banner », « main » et « contentinfo » ne peuvent être présents qu'une seule fois par page. Il en va de même pour les balises html <header>, <main> et <footer>.

Le nombre de landmarks par page devrait être limité à l'essentiel pour faciliter la navigation par le lecteur d'écran.
Il n'existe pas de limites maximum au nombre de landmarks mais leur multiplication entraine une lourdeur de lecture. Il est recommandé de se limiter aux landmarks essentiels appropriés à leur contenu.

Les niveaux de titres (headings)

Les niveaux de titres (1 à 6) jouent un rôle essentiel pour l'accessibilité d'un site ou d'un document (Word, PDF…). Ils communiquent à tout utilisateur, une structure claire de la page. Le squelette ainsi créé permet aux utilisateurs de lecteurs d'écran de voir une vue synthétique du contenu de la page et de naviguer vers le titre souhaité.

Une bonne structure de titres améliore par ailleurs le référencement dans les moteurs de recherche et facilite la maintenance et les changements de design d'un site. Dans un document Word, ils permettent par ailleurs la génération automatique d'une table des matières.

a) De vrais niveaux de titres

Les textes qui servent de titres visuels ou structurels doivent utiliser les bonnes balises de titres
Les titres sont utilisés pour aider l'utilisateur à naviguer dans une page. Le but est sémantique et non visuel. Un lecteur d'écran ne peut lire le CSS et ne pourra déduire qu'un texte plus grand ou en gras doit être considéré comme un titre.

Les titres seront donc balisés dans des balises <h1> à <h6> et les niveaux de <hx> ne sont pas choisis en fonction du design mais de la hiérarchisation des titres. Le CSS pourra, par la suite, modifier le design de chaque niveau de titre.

Bon exemple : De vrais titres correctement balisés

```
<h1>Types of Web Accessibility Laws</h1>

<h2>Civil Rights Laws</h2>

<h2>Procurement Laws</h2>

<h2>Industry-Specific Laws</h2>
```

Bon exemple : De vrais titres balisés avec ARIA

Dans certains cas de « réparation d'accessibilité » d'un site, ARIA peut être utilisé pour transformer un composant en niveau de titre. Le role=« heading » combiné à aria-level a été créé à cet effet.

Attention, il est toujours préférable d'utiliser les balises natives <h1> à <h6> et ARIA est une solution de secours (même robuste).

```
<div role="heading" aria-level="1">

    Screen readers will recognize this as a heading level 1
```

```
</div>
```

Mauvais exemple : De faux titres qui utilisent des styles

Les paragraphes <p style…> ci-dessous ressembleront visuellement à des titres sans en avoir la fonction.

Notons par ailleurs que les utilisateurs qui ont besoin d'une autre taille, couleur, police… adaptées à leur profil ne pourront pas les adapter facilement si le style est intégré au html comme dans cet exemple.

```
<div id="noHeadingExample">
<p style="font-weight: bold; font-size: 200%;">
  Setting the Exposure Manually on a Camera
</p>
  <p>Put text here...</p>
  <p style="font-weight: bold; font-size: 150%;">
    Set the ISO
  </p>
    <p>Put text here...</p>
  <p style="font-weight: bold; font-size: 150%;">
    Choose an aperture
  </p>
    <p>Put text here...</p>
  <p style="font-weight: bold; font-size: 150%;">
    Choose a shutter speed
  </p>
      <p>Put text here...</p>
</div>
```

Les textes qui n'ont pas vocation à servir de titres ne peuvent pas être balisés par des niveaux de titres <hx>

Les textes qui nécessitent un design particulier sans avoir une fonction de titre doivent être designé en CSS sans utiliser de balise de titre. Une telle pratique rend la lecture confuse pour les utilisateurs de lecteurs d'écran et peut générer des erreurs ou des incompréhensions.

Mauvais exemple : Un texte d'accroche promotionnel balisé comme un titre

Special rates good until July 31!

```
<h1 class="promotional-item">Special rates good until July 31!</h1>
```

b) Des titres significatifs explicites

Les titres doivent être appropriés et informatifs

Les titres doivent être suffisamment clairs pour permettre à l'utilisateur de comprendre le sens du texte qu'il introduit. Le lecteur d'écran lit à l'utilisateur le niveau de titre puis le contenu du <hx>.

Il n'existe pas de limite de nombre de caractères mais il est recommandé d'être le plus court et explicite possible. Le but est d'introduire brièvement le contenu concerné qui suit le titre.

Bon exemple : Des titres explicites

<h1>Emergency Preparedness Guide</h1>

<h2>Know the Risks</h2>

<h2>Make a Plan</h2>

<h2>Get an Emergency Kit</h2>

<h3>Emergency Kit Basic Items</h3>

<h2>Resources<h2>

Mauvais exemple : Des titres génériques

<h1>Emergency Preparedness Guide</h1>

<h2>Section 1</h2>

<h2>Section 2</h2>

<h2>Section 3</h2>

<h3>(a)</h3>

<h2>Section 4<h2>

c) Hiérarchie et structure du contenu

Les niveaux de titres doivent présenter une structure cohérente de l'organisation des sections de la page

Penser toujours d'abord à la structure de la page et s'occuper du design en second lieu. Le contenu est prioritaire et le CSS est géré après. Cette démarche sera par ailleurs bénéfique au référencement et permet de faciliter les mises en page sur le mobile (cacher ou agencer des contenus en fonction des besoins).

Si la structure d'une page ne contient que des titres avec peu de texte, il faut peut-être envisager de remplacer la structure par des listes à puces. Par contre, si le contenu prévoit un ou des paragraphes d'explications, une structure et une hiérarchie de titres doivent être envisagées pour guider l'utilisateur vers le contenu qui l'intéresse et lui montrer rapidement la « table des matières » de la page.

Bon exemple : Une structure logique de titres pour régler l'exposition d'un appareil photo

Une liste d'étapes sans contenu peut se rédiger dans une liste à puces

 Setting the Exposure Manually on a Camera


```
                    <li>Set the ISO</li>
                    <li>Choose an aperture</li>
                    <li>Set a shutter speed</li>
                </ul>
            </li>
            <li>…</li>
</ul>
```

Mais si chaque étape nécessite davantage d'explications, une structure de titres est toute indiquée

```
<h1>Setting the Exposure Manually on a Camera</h1>
    <p>Put text here...</p>
  <h2>Set the ISO</h2>
    <p>Put text here...</p>
  <h2>Choose an aperture</h2>
    <p>Put text here...</p>
  <h2>Choose a shutter speed</h2>
    <p>Put text here...</p>
```

Commencer le contenu de chaque page par un <h1> et se limiter à un seul <h1> par page

Les lecteurs d'écran intègrent des raccourcis qui permettent de naviguer directement au niveau de titre souhaité. Si la page ne commence pas par un <h1>, l'internaute risque d'être perdu. Il en va de même pour une page qui proposerait plusieurs <h1>.

Pour ces raisons, chaque contenu de page devrait commencer par un <h1>. Sauf exceptions (fenêtre modale interactive, liste d'articles sur la page…), il ne faudrait prévoir qu'une balise <h1> unique sur la page. Ce <h1> devrait idéalement contenir du texte dont le sens est proche du <title> de la page.

Cette pratique participera également à un meilleur positionnement dans les moteurs de recherche et permettra des mises à jours cosmétiques plus rapides.

d) Il ne peut pas y avoir de cassures dans la hiérarchie des niveaux de titres

La hiérarchie des niveaux de titre doit être respectée. On ne passera par exemple pas d'un niveau <h3> à un <h5> sans prévoir un <h4> intermédiaire. Cette erreur est souvent commise par choix d'un niveau de titre sur base du design et non pas sur base de la sémantique. Comme déjà écrit plus haut, penser d'abord le contenu et le design se gère par la suite en CSS.

La hiérarchie du contenu peut se comprendre comme un arbre généalogique :

Tout en haut, l'arrière- grand-mère <h1>

…qui a des enfants (grands-parents) <h2>

…qui ont des enfants (parents) <h3>

…qui ont des enfants (enfants) <h4>

…qui ont des enfants (petits-enfants) <h5>

…qui ont des enfants (arrière-petits-enfants) <h6>.

Dans cet arbre, un parent (h3) ne peut pas se trouver au même niveau qu'un petit-enfant (h5) et oublier un niveau <hx> aurait le même impact que de passer du niveau arrière-grand-mère (h1) au niveau parents (h3), ce qui est impossible puisque les grands-parents sont les géniteurs des parents…

Bon exemple : Les étapes détaillées des réglages de l'exposition d'un appareil photo

```
<div class="exampleHeadings">

<h1>Setting the Exposure Manually on a Camera</h1>

  <h2>Setting the ISO</h2>

    <h3>The effect of ISO on image quality</h3>

      <p>The higher the ISO, the greater your ability to take photos in low light [...]</p>

    <h3>High ISO cameras</h3>

      <p>Cameras with larger sensors are capable of higher ISO values [...]</p>

  <h2>Choose an aperture</h2>

    <h3>The effect of aperture on depth of focus</h3>

      <p>The larger the aperture, the narrower the plane of focus [...]</p>

    <h3>Vignetting</h3>

      <p>Vignetting occurs when a lens lets in less light around the edges than in the center [...]</p>

    <h3>Diffraction</h3>

      <p>A small aperture can cause the light to diffract, which reduces the sharpness of the image, even though more of the image is in focus [...]</p>

  <h2>Choose a shutter speed</h2>

    <h3>Shutter speed limitations for hand-holding a camera</h3>

      <p>Hand-holding a camera at a slow shutter speed increases the chance of vibration and blur in the final image [...]</p>

    <h3>Long exposures</h3>

      <p>Use a tripod when taking long exposures [...]</p>

</div>
```

Mauvais exemple : Un plan déstructuré

L'exemple ci-dessous présente deux problèmes :

1. La page commence par deux titres <h3> plutôt qu'un <h1> et aucun de ces titres ne peut être considéré comme titre principal de la page.

2. Plus bas, le contenu contient un <h1> qui passe directement à un <h3>

```
<h3>Quick Links</h3>
  <h3>Photography Tutorials</h3>
<h1>Setting the Exposure Manually on a Camera</h1>
  <h3>Set the ISO</h3>
    <h4>The effect of ISO on image quality</h4>
    <h4>High ISO cameras</h4>
  <h3>Choose an aperture</h3>
    <h4>The effect of aperture on depth of focus</h4>
    <h4>Vignetting</h4>
    <h4>Diffraction</h4>
  <h3>Choose a shutter speed</h3>
    <h4>Shutter speed limitations for hand-holding a camera</h4>
    <h4>Long exposures</h4>
```

Les liens

Les lecteurs d'écran reconnaissent les liens et les annoncent comme tels. Il n'est donc pas nécessaire d'ajouter, par exemple, les mots « lien vers… ».

a) Désigner correctement les liens

Les liens doivent être sémantiquement désignés comme des liens

Les liens doivent se trouver dans une balise de lien <a href…> ou désignés comme liens grâce à ARIA.

Bon exemple : Un lien dans une balise <a> qui contient une valeur valide

```
<a href="https://eqla.be">Eqla</a>
```

Bon exemple : Un lien désigné comme tel avec ARIA qui contient une valeur valide

Il est préférable d'utiliser la balise <a> pour créer des liens mais dans certains cas (réparations de sites, corrections de défauts d'accessibilité…) il est possible de transformer l'élément qui contient le lien en un élément de type lien grâce à ARIA.

Dans ce cas, les 3 éléments suivants doivent être réunis :

- Ajout du role=« link » sur l'élément
- Ajouter l'attribut tabindex=« 0 » sur l'élément pour qu'il prenne le focus en navigation au clavier
- S'assurer que le script associé en JavaScript permette d'activer le lien à la souris ET au clavier.

```
<span role="link" tabindex="0" data-href="https://eqla.be">Eqla</span>
```

Mauvais exemple : Un faux lien dans un spam créé en JavaScript

Le code ci-dessous va afficher un lien vers le site d'Eqla dans la page. Le texte ressemble à un lien, fonctionne à la souris mais n'est pas considéré comme un lien par les technologies d'assistance et ne prend pas le focus.

```
<script>

function openInNewTab(url) {

  var win = window.open(url, '_blank');

  win.focus();

}

</script>

<span class="fakeLink" onclick="openInNewTab('https://www.eqla.be');">Eqla</span>
```

Les liens et les boutons devraient être utilisés sémantiquement pour la fonction pour laquelle ils ont été créés

Les lecteurs d'écran spécifient à leurs utilisateurs le type d'élément sur lequel ils se trouvent. Ils commencent par dire « lien » ou « bouton » à la prise de focus.

Utiliser un lien pour un bouton ou l'inverse peut engendrer des incompréhensions ou des erreurs de manipulations pour les utilisateurs concernés. Par exemple, si l'internaute liste spécifiquement les liens d'une page, ceux considérés comme des boutons ne seront pas listés.

La documentation HTML renseigne bien les cas d'usage que nous nous permettons de rappeler ici :

1. Les liens pointent vers une autre page ou un emplacement spécifique sur la page
2. Les boutons activent une fonctionnalité en général sur une même page (ouverture d'une boîte de dialogue, ouvrir un contenu dans un accordéon…) ou envoient des données à la fin d'un formulaire.

b) Le texte des liens

Un lien doit contenir techniquement un texte qui sera reconnu comme nom accessible (accessible name)

Un lien doit être associé à un texte qui servira de nom accessible calculé par l'accessibility tree lors du calcul algorithmique. En arrivant sur un lien, le lecteur d'écran lit « lien » puis le nom accessible. Si ce nom n'est pas présent ou incorrectement implémenté, le lecteur d'écran lira le contenu du « href » pour tenter de donner une information à l'utilisateur. Ce contenu non explicite s'avèrera souvent trop long, incompréhensible ou pourra induire en erreur.

Le nom accessible est calculé dans l'ordre de préférence suivant par les lecteurs d'écran :

1. Aria-labelledby
2. Aria-label
3. Texte contenu entre la balise d'ouverture <a> et de fermeture incluant l'attribut alt des images
4. L'attribut title. Ce cas est à considérer en tout dernier recours car il peut générer d'autres défauts d'accessibilité.

Même si le texte contenu entre les balises <a> apparaît en troisième position, il devrait être la méthode à utiliser en premier car facile à mettre en place, nativement prévu en HTML, non source d'erreur dans les mises à jour du site par un webmaster non formé à cette technique

qui ignorerait cette méthode de calcul sans se rendre compte que son texte de lien n'est pas correctement compris.

Bon exemple : Un lien sous forme de texte dans une balise <a>

```
<p><a href="https://eqla.be/contact/">Contact Us</a></p>

<p><a href="https://eqla.be/accueil/notre-equipe/">Our team</a></p>
```

Bon exemple : Un lien sous forme de texte dans l'attribut alt d'une image-lien

```
<p>
  <a href="https://eqla.be/">
    <img src="eqla-logo.png" alt="Home - Eqla">
  </a>
</p>
```

Bon exemple : aria-label pour écraser le texte natif du lien

Cet exemple montre comment aria-label peut rendre explicite un texte de lien générique du type « Read more… » utilisé régulièrement sous forme de « teaser » dans une page qui liste plusieurs publications.

Aria-label remplace le texte d'origine « Read more » par « Read more + le titre de la publication visée »

```
<p>
  The National Museum of African American History and Culture
  was established in 2003 by an Act of Congress, making it the
  19th Smithsonian Institution museum.
    <a href="https://www.si.edu/Museums/african-american-history-and-culture-museum"
      aria-label="Read more about the National Museum of African American History and Culture">
        Read more...
    </a>
</p>
<p>
  The National Air and Space Museum, Steven F. Udvar-Hazy Center displays hundreds of aviation and space artifacts that are too large to exhibit in the museum on the National Mall in Washington, D.C.
    <a href="https://www.si.edu/Museums/air-and-space-museum-udvar-hazy-center"
      aria-label="Read more about the Steven F. Udvar-Hazy Center">
        Read more...
    </a>
</p>
```

Bon exemple : Du texte complémentaire dans un lien caché à l'écran mais lisible par les lecteurs d'écran

Dans certains cas, il peut être utile de donner un complément d'information uniquement aux utilisateurs de lecteurs d'écran via du texte caché en CSS grâce à la méthode clip.

```
<head>
  <title>Museum Information</title>
  <style>
    .visually-hidden {
      position: absolute;
      clip: rect(0 0 0 0);
      border: 0;
      height: 1px; margin: -1px;
      overflow: hidden;
      padding: 0
      width: 1px;
      white-space: nowrap;
    }
  </style>
</head>
<body>
<p>
  The National Museum of American History is devoted to the scientific,
  cultural, social, technological, and political development of the United States.
    <a href="https://www.si.edu/Museums/american-history-museum">
      Read more
      <span class="visually-hidden">
        about the National Museum of American History
      </span>...
    </a>
</p>
</body>
```

Bon exemple : Utiliser aria-label pour donner un texte de lien aux images en background

Les images background associées à un lien sont souvent utilisées pour faire un lien vers les réseaux sociaux. Cette technique fonctionne pour les personnes voyantes mais ne permet pas de donner un nom accessible au lien.

Dans cet exemple, aria-label fournit un texte au lien que les lecteurs d'écran pourront communiquer aux utilisateurs.

<a href=" https://www.facebook.com/Eqla.asbl/" class="facebook"

aria-label="Page Facebook d'Eqla">

L'objet de chaque lien devrait être explicite directement grâce au texte du lien

Le contexte du lien est constitué de ce qui entoure directement le lien : Un titre, un paragraphe…

Les utilisateurs de lecteurs d'écran peuvent découvrir le contenu d'une page en lisant tout son contenu de haut en bas ou en passant de titre en titre, de lien en lien… Lors d'une navigation de lien en lien, même si le contexte du lien est explicite, un texte de lien générique comme « lire plus » demandera davantage de manipulations au lecteur pour comprendre l'objet du lien.

Aussi, la technique la plus facile et robuste consiste à rendre le lien explicite en dehors de son contexte en renseignant son objet directement dans le texte du lien.

Sont visés ici des liens génériques courts du type « Cliquer ici », « Ici », « Lire plus », « Plus »…

Les liens explicites présentent par ailleurs d'autres avantages :

- Un lien riche de sens participe à l'amélioration du référencement naturel (SEO)
- Une surface de clic plus grande est plus facilement atteignable par des personnes malvoyantes ou dont les capacités à viser une petite cible sont diminuées
- Une surface de clic plus grande invite l'utilisateur à « passer à l'action » et à consulter davntage de pages sur le site.

Bon exemple : Des liens qui ont du sens même hors contexte

Learn more about our products.

Read a fascinating article about the resident microbes in the human body.

Our second quarter earnings have surpassed investor expectations.

Mauvais exemple : Des liens dont le texte n'est pas descriptif

Learn more about our products here.

To read a fascinating article about the resident microbes in the human body click here.

Our second quarter earnings have surpassed investor expectations. More...

Les fonctionnalités telles que les labels, names, et les textes alternatifs qui ont la même fonction doivent être identifiés de la même manière

Le but est d'éviter l'équivoque et de rester cohérent sur l'ensemble du site

Mauvais exemple : Des liens vers la même page non décrits par le même texte

```
<p><a href="contact.html">Our Company</a></p>

<p><a href="contact.html">Contact Us</a></p>
```

Mauvais exemple : Des liens vers des pages différentes décrits par le même texte

```
<p><a href="contact.html"></a>Contact Us</a></p>

<p><a href="directory.html">Contact Us</a></p>
```

c) Liens vers sites externes, nouvelles fenêtres et fichiers

Un lien qui ouvre une nouvelle fenêtre ou onglet devrait indiquer cette ouverture

Tous les utilisateurs devraient être avertis de l'ouverture du lien dans une nouvelle fenêtre soit par du texte du type « nouvelle fenêtre », soit par un indice visuel doublé d'une alternative textuelle (alt). Cette indication doit être visible et comprise par les internautes voyants et non voyants.

Mauvais exemple : Pas d'indication de l'ouverture dans une nouvelle fenêtre

```
<p>

 <a href="https://eqla.be" target="_blank">Eqla</a>

</p>
```

Bon exemple : Une image avec une alternative qui indique l'ouverture dans une nouvelle fenêtre

Les personnes voyantes voient l'image et les non-voyants entendent l'alternative de l'image.

```
<p>

 <a href="https:// eqla.be" target="_blank">Eqla

  <img src="newWindow.png" alt="opens new window" width="16" height="16">

 </a>

</p>
```

Bon exemple : L'utilisation de CSS et ARIA pour indiquer l'ouverture dans une nouvelle fenêtre

Ici aussi, les personnes voyantes voient l'image et les non-voyants entendent l'alternative de l'image.

```
<p>

 <a aria-describedby="a11y-message--new-window" class="icon--new-window" href="https://eqla.be" target="_blank">Eqla</a>

</p>
```

.

.

 (opens new window)

La class icon—new--window est utilisée pour injecter l'icone via CSS (glyph icon) après le texte du lien.

Le qui contient l'id=« a11y-message--new-window » fournit le texte « opens a new window » repris une seule fois sur la page et caché à l'écran en CSS avec la méthode clip et caché aux lecteurs d'écrans avec aria-hidden="true".

Bon à savoir : aria-describedby et aria-labelledby peuvent accéder au contenu présent dans un conteneur caché avec aria-hidden="true". Dans notre exemple, cela signifie que le message « opens a new window » sera lu lors de la prise du focus du lien au clavier mais il empêche les utilisateurs de lecteurs d'écran d'entendre le message s'ils lisent le contenu de la page en mode lecture (pas de tabulation).

L'avantage de cette technique par rapport à l'exemple précédent (image + alt) est le chargement de l'image une seule fois sur la page. L'inconvénient est que NVDA et JAWS ne lisent pas le message « opens a new window » en mode lecture sans tabulation avec prise de focus sur le lien.

Un lien vers un fichier, une application ou un format autre que web devrait indiquer le type de fichier ou d'application, son poids et sa langue.
Tous les utilisateurs devraient connaître à l'avance le type de fichier ou l'application qui sera ouvert au clic sur le lien. L'indication peut être communiquée en texte (ex : pdf, xlsx, docx…) ou via une image qui contient une alternative textuelle comme illustré plus haut dans le cas d'ouverture d'une nouvelle fenêtre.

Par ailleurs, pour éviter toute mauvaise surprise, le lien contiendra également le poids du document (ex : 2MO) et la langue du document s'il est écrit dans une autre langue que celle de la page (ex : EN pour un document en anglais sur une page web en français).

Ces bonnes pratiques permettent à tous les utilisateurs de gagner du temps en ne téléchargeant que des documents utilisables sur leur appareil (pas de lecteur .xlsx sur mon smartphone), sans exploser leur forfait data (2 GO de fichier…) et dans une langue connue.

Bon exemple : Télécharger notre rapport d'activité
Télécharger le rapport d'activité 2022 d'Eqla (PDF, 2MO, EN)

d) Visualisation des liens

Les liens doivent pouvoir être distingués visuellement dans leur contexte environnant
Par défaut, les liens en HTML sont parfaitement reconnaissables du texte environnant de par leur couleur et le soulignement mais ce comportement par défaut est tout logiquement modifié pour coller à la charte graphique du site.

La couleur seule ne peut être utilisée pour distinguer un lien du reste du texte sauf si le ratio de contraste respecte les standards d'accessibilité (au moins 3 :1) et qu'une autre différentiation est appliquée lorsque le lien est survolé (hover) ou lorsqu'il prend le focus.

Il existe plusieurs moyens accessibles de différencier les liens du texte environnant :

- Une couleur différente + soulignement (comportement standard).
- Une couleur différente + soulignement au survol ET au focus. Ce comportement est souvent désactivé par certains designers qui n'aiment pas ce style.

- Une couleur de background différente au survol ET au focus + soulignement au survol ET au focus.
- Un outline ou une bordure au survol ET au focus + soulignement ET/OU couleur de background différent au survol ET au focus.
- Menu de navigation : dans le cas des menus de navigation, les utilisateurs s'attendent à trouver uniquement des liens vers d'autres pages ou sur la même page et le soulignement n'est pas nécessaire.

e) Indication visuelle du focus

Tous les éléments qui prennent le focus doivent avoir une indication visuelle à la prise de focus

Par défaut, les navigateurs appliquent une bordure de couleur continue ou pointillée sur un composant qui prend le focus. La bordure continue est plus visible que la pointillée qui correspond cependant aux standards minimum d'accessibilité.

Malheureusement, soit volontairement, soit à cause d'un reset CSS mal conçu, certains designs de sites dégradent cette indication visuelle.

Dès lors, les utilisateurs voyants qui utilisent d'autres moyens de navigation que la souris ne peuvent se situer sur le site Internet.

Mauvais exemple : L'indication de focus est supprimée

```
<style>
  a:focus {
    outline: none;
  }
</style>
```

Les éléments qui prennent le focus devraient bénéficier d'un indice visuel améliorer

Pour permettre à tous les utilisateurs de naviguer plus facilement sur un site, l'indication visuelle de focus par défaut devrait être renforcée.

Il existe plusieurs moyens de renforcer le focus des liens, boutons, éléments de formulaires et autres éléments. Le renforcement peut être appliqué sur le background, la couleur de police ou la bordure :

- Liens (links) – a:focus {…}
- Boutons (buttons) – button:focus {…}
- Champs de formulaires (inputs) – input:focus {…} pour tous les inputs mais il est possible de styler les types d'inputs comme suit :
  - text inputs — input[type=text]:focus {...}
  - image inputs — input[type=image]:focus {...}
  - submit buttons — input[type=submit]:focus {...}
  - radio buttons — input[type=radio]:focus {...}
- Cases à cocher (checkboxes) — input[type=checkbox]:focus {...}
- Menus déroulants (drop-down) selection inputs — select:focus {...}
- Champs de commentaires (textarea fields) — textarea:focus {...}
- ARIA controls:
- ARIA links — [role=link]:focus {...}
- ARIA buttons — [role=button]:focus {...}

- ARIA inputs — [role=radio], [role=checkbox], etc.
- ARIA tabs — [role=tab]:focus {...}

Bon exemple : L'indication de focus est renforcée

```
<style>
  a:focus {
    outline: 2px solid #8cc63f;
    background-color: #fdf6e7;
  }
</style>
```

Navigation entre plusieurs pages

Les menus de navigation devraient être balisés par la balise <nav> ou un role=« navigation ». Le nombre et l'ordre de liens dans le menu devraient être cohérents et identiques sur toutes les pages.

a) Listes de navigation

Une liste de navigation devrait être désignée par la balise <nav> ou le role= « navigation »

La navigation principale devrait être balisée par <nav> ou role= « navigation ». Il n'est pas souhaitable de baliser toutes les navigations secondaires de cette manière ou, afin de ne pas perdre l'utilisateur, il y a lieu de donner un label explicite aux différentes listes de navigation (voir chapitre sur les régions).

Bon exemple : utilisation de la balise <nav>

```
<nav>
  <ul>
    <li><a href="home/">Home</a></li>
    <li><a href="home/products">Products</a></li>
    <li><a href="home/services">Services</a></li>
  </ul>
</nav>
```

Bon exemple : utilisation du role= « navigation »

Attention à bien assigner le role au div et non au <ul> pour ne pas écraser la sémantique de la liste à puces et permettre au lecteur d'écran de préciser qu'il s'agit d'une liste à puces qui contient x éléments.

```
<div role="navigation">
  <ul>
    <li><a href="home/">Home</a></li>
    <li><a href="home/products">Products</a></li>
    <li><a href="home/services">Services</a></li>
```

```
  </ul>
</div>
```

Une liste de navigation devrait informer visuellement les utilisateurs sur la page en cours dans le menu

Le but est de mettre en évidence, dans le menu, la page active. La règle est la même pour un ensemble de pages qui appartiennent à la même collection comme, par exemple, les résultats d'un moteur de recherche dans plusieurs pages. La page en cours devrait être identifiée visuellement dans la liste des pages (1, 2, 3…).

Bon exemple : Mise en évidence de la page en cours

La page en cours est identifiée grâce à une class « current-page »

```
<head>
  <title>Groceries</title>
  <style>
    li.current-page {
      outline: red solid 1px;
    }
  </style>
</head>
<body>
        <nav>
        <ul>
                <li><a href="#">Fruits</a></li>
                <li><a href="#">Vegetables</a></li>
                <li><a href="#">Meats</a></li>
                <li class="current-page">Dairy</li>
                <li><a href="#">Breads, Pasta, & Cereals</a></li>
                <li><a href="#">Soups & Canned Goods</a></li>
                <li><a href="#">Frozen Foods</a></li>
                <li><a href="#">Desserts</a></li>
                <li><a href="#">Snack Foods</a></li>
        </ul>
        </nav>
</body>
```

Une liste de navigation devrait permettre aux utilisateurs de lecteur d'écran d'identifier la page en cours

L'attribut aria-current indique aux lecteurs d'écran l'élément courant, comme par exemple la page active. Cet attribut est supporté par la plupart des combinaisons lecteurs d'écran / navigateurs.

La règle est la même pour un ensemble de pages qui appartiennent à la même collection comme, par exemple, les résultats d'un moteur de recherche dans plusieurs pages. La page en cours devrait être identifiée par les technologies d'assistance dans la liste des pages (1, 2, 3…).

Bon exemple : identification de la page en cours avec aria-current

```
<ul>
    <li><a href="#">Fruits</a></li>
    <li><a href="#">Vegetables</a></li>
    <li><a href="#">Meats</a></li>
    <li><a href="#" aria-current="page">Dairy</a></li>
    <li><a href="#">Breads, Pasta, & Cereals</a></li>
    <li><a href="#">Soups & Canned Goods</a></li>
    <li><a href="#">Frozen Foods</a></li>
    <li><a href="#">Desserts</a></li>
    <li><a href="#">Snack Foods</a></li>
</ul>
```

Cohérence de la navigation sur toutes les pages du site

Les éléments du menu et leur ordre doivent être identiques sur toutes les pages du site

Les utilisateurs pourraient être déstabilisés par un changement de menu d'une page à l'autre et perdront du temps à comprendre les changements de navigation.

Navigation au sein de la page

Les utilisateurs de lecteurs d'écran comprennent la structure de la page grâce au balisage sémantique. Ils peuvent naviguer facilement et de plusieurs façons entre les éléments de même type… mais certaines fonctionnalités ne sont pas incluses nativement comme par exemple le « vous êtes ici » dans les résultats de recherches.

Par ailleurs, les personnes voyantes utilisatrices du clavier seulement sans lecteur d'écran n'ont pas accès à ces fonctionnalités et il est nécessaire de leur fournir des moyens de naviguer plus facilement et rapidement.

a) Les skip links ou liens d'évitement

Une fonctionnalité de liens d'évitement devrait permettre aux utilisateurs de naviguer directement dans le contenu principal sur chaque page

Avant d'arriver dans le contenu principal <main> de la page, l'utilisateur doit passer par une série d'étapes comme le logo de l'entête, le moteur de recherche, le widget de login, la navigation principale (qui peut parfois être très longue)… Dans certains cas, une navigation secondaire est également positionnée avant le contenu principal. Les utilisateurs devraient

pouvoir passer tous ces contenus en grâce à un raccourci sous forme de lien d'évitement positionné en premier lieu sur la page et accessible au clavier.

Quelques précautions d'usage :

- Le skip link est le premier élément focusable sur la page
- Il doit être positionné au tout début du document juste après l'ouverture du body
- Le lien pointe vers l'id qui identifie le début du contenu principal. Il devrait s'agir du <main> ou du <h1>
- Le skip link doit donner le focus à l'élément cible et pas seulement faire défiler l'écran jusqu'au contenu à consulter.

Bon exemple de skip link

```
<body>

<div id="skipnav"><a href="#mainContent">Skip navigation</a></div>

    <!-- the header, navigation, etc. go here -->

<main id="mainContent" tabindex="-1">

    <!-- the main content goes here -->

</main>

…

</body>
```

Certains navigateurs, comme Safari, nécessitent un élément de destination focusable nativement (lien, bouton…) ou qui contient une valeur de tabindex. Sinon, le viewport scrollera vers la position voulue mais la tabulation suivante positionnera le focus sur l'élément qui suit directement le lien d'évitement.

L'ajout d'un tabindex= « -1 » sur l'élément de destination y positionne le focus sans interférer avec l'ordre éventuel de tabulation présent sur la page.

Mauvais exemple : Pas de skip link sur un site de vente qui contient beaucoup d'étapes avant le main

Les images ci-dessous illustrent le site Walmart qui contient approximativement 76 liens et 5 champs de formulaire avant d'arriver dans le contenu principal. Même si quelques liens sont cachés via Javascript, le nombre de tabulations nécessaire à chaque ouverture de page est laborieux et rend la consultation très longue.

La copie d'écran où le CSS est désactivé montre la séquence linéaire telle qu'elle est parcourue par un lecteur d'écran ou via la tabulation au clavier.

Un skip link doit être visible soit en permanence soit à la tabulation clavier.
L'option la plus accessible à tous est de laisser le lien d'évitement visible pour tous en permanence mais, pour des raisons de charte graphique ou de préférences en matière de design, le lien peut est caché visuellement tout en restant lisible par les lecteurs d'écran et activable à la tabulation clavier.

Le lien d'évitement ne doit pas être rendu invisible à l'écran en display :none car ce paramètre rendrait le lien inutilisable au clavier et pour les lecteurs d'écran. Il est recommandé d'utiliser la méthode clip en CSS déjà abordée quelques fois dans ce document.

Bon exemple de lien d'évitement caché et visible au focus

```
<head>
  <title>Museum Information</title>
  <style>
    #skipnav a {
      position: absolute;
      clip: rect(0 0 0 0);
      border: 0;
      height: 1px; margin: -1px;
      overflow: hidden;
      padding: 0
      width: 1px;
      white-space: nowrap;
```

```
    }
    #skipnav a:focus {
      clip:auto;
      left:0;
      top:0;
      width:100%;
      height:auto;
      margin:0;
      padding:10px 0;
      background:#fdf6e7;
      border:2px solid #990000;
      border-left:none;
      border-right:none;
      text-align:center;
      font-weight:bold;
      color:#990000;
    }
  </style>
</head>
<body>
  <div id="skipnav"><a href="#mainContent">Skip navigation</a></div>
  <!-- document banner, navigation, etc. -->
  <main id="mainContent" tabindex="-1">
      <h1>Link will take users to this location.</h1>
      <!-- other content in the main content -->
  </main>
  <!-- other content on the web page -->
</body>
```

b) Autres considérations en navigation interne

Table des matières

Une table des matières en début de page améliore la compréhension du son contenu et facilite la navigation, notamment par les personnes voyantes qui ne peuvent utiliser la souris. La table des matières donne un aperçu concis des sujets abordés dans la page et l'ancre permet de naviguer directement à la partie souhaitée.

Si une table des matières est prévue, elle devrait refléter la structure de titres de la page. En fonction des cas, on choisira de lister tous les niveaux de titres ou uniquement les <h2>, <h2> + <h3>…

Ordre de tabulation et de consultation

Le lecteur d'écran lit la page dans l'ordre des contenus dans le DOM donc, en théorie, ce qui est visible à l'écran correspond à l'ordre de lecture par les personnes non voyantes, mais il faut être attentif aux positionnements en CSS ou à des injections dynamiques, par exemple en AJAX, qui peuvent modifier l'ordre d'apparition.

La lecture du contenu doit être logique et intuitive. La désactivation du CSS est une bonne technique pour visualiser l'ordre d'apparition des contenus.

La navigation au clavier via la touche de tabulation doit également être logique. La navigation doit commencer en haut de la page puis descendre sans passer d'étape dans le sens logique de la lecture.

Les tabindex avec une valeur positive ne devraient pas être utilisés. D'abord parce qu'ils peuvent modifier l'ordre logique de tabulation et créer des confusions. Ensuite, parce qu'ils suppriment l'élément de l'ordre naturel de tabulation et le placent en premier dans l'ordre de tabulation. Un tabindex avec une valeur supérieure ou égale à 1 deviendra prioritaire sur tous les autres éléments. Les comportements da la tabulation au clavier peut s'en trouver très perturbée et perdre complètement l'utilisateur.

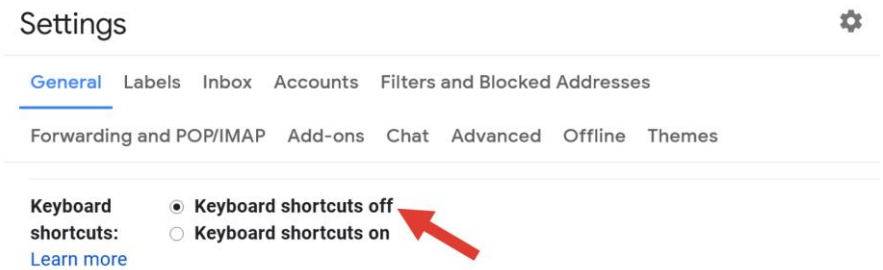
Raccourcis à touche unique

Les applications de l'utilisateur et ses propres paramètres, liés ou non à une technologie d'assistance, utilisent peut-être des raccourcis clavier.

Certains internautes, de par certaines limitations de mouvements, commettent beaucoup d'erreurs de frappe et peuvent activer des touches de manière non intentionnelle.

Dès lors, si un raccourci à touche unique existe sur le site, l'utilisateur doit pouvoir le désactiver ou le modifier.

Par exemple, Gmail utilise beaucoup de raccourcis à touche unique (« e » pour archiver une conversation, « a » pour répondre à tous, « f » pour forwarder…) mais permet à l'utilisateur de paramétrer ces options.



Les listes

En arrivant sur une liste, les lecteurs d'écrans annoncent la liste, le début et la fin de la liste et le nombre d'éléments contenus. Ce comportement est très utile pour les personnes aveugles qui ont ainsi une vue d'ensemble des éléments, tout comme un personne voyante.

a) Les listes doivent être construites en utilisant la bonne sémantique HTML

En fonction des usages, les listes peuvent être :

1. Des listes non ordonnées <ul> qui regroupent des éléments dans des <li>. À utiliser pour lister une suite d'éléments qui peuvent être placés dans n'importe quel ordre.
2. Des listes ordonnées <ol> qui regroupent des éléments dans les <li>. À utiliser pour lister des éléments dans un ordre précis.
3. Des listes de définitions <dl> qui regroupent des termes <dt> liés à leur définition <dd>.

Les lecteurs d'écran comprennent ces 3 types et lisent le contenu en toute cohérence.

L'usage de
 et / ou de caractères (tiret, étoile…) au clavier pour créer des fausses listes ne permettra pas à l'utilisateur de lecteur d'écran de comprendre la logique entre les éléments. La lecture sera par ailleurs alourdie et parfois incompréhensible. La
 pourra être vocalisé « passage de ligne », le tiret « tiret »…

Le respect d'un bon balisage de listes aura un impact positif sur le référencement et facilitera la cohérence du design des pages sur le site.

b) Les balisages de listes ne doivent pas être utilisés pour d'autres contenus que des listes.

Pour des raisons de design, par manque de connaissances, par « facilité », les puces sont parfois utilisées pour attirer l'attention de l'utilisateur de manière visuelle (une puce qui représente une flèche de direction…).

Cet usage entraîne une lourdeur de lecture et peut engendrer des incompréhensions.

À retenir :

- Des listes uniquement pour des besoins de sémantique et pas pour le design
- Des listes regroupent des éléments d'une même famille
- Utilisation d'une liste à partir de plus de deux éléments de la même famille

c) Éviter les imbrications de listes supérieures à 2 niveaux

Il est possible d'imbriquer des listes dans des listes. La lecture et la compréhension peuvent devenir compliquées à partir de deux imbrications.

Dans un tel cas, il est préférable de revoir la structure de la page et d'utiliser des niveaux de titres supplémentaires.

Les cadres (Iframes)

Les cadres incluent du contenu tiers dans une page web. Ce contenu peut être de la publicité, des vidéos, des publications de réseaux sociaux, des widgets, des formulaires… Dans certains cas, les cadres contiennent du contenu JavaScript caché aux utilisateurs.

Du point de vue de l'accessibilité, les Iframes doivent contenir un title pour permettre aux utilisateurs de naviguer entre eux, de les trouver et de les identifier. Il est important de savoir également que les éléments du cadre (niveaux de titres, liens…) seront compris par les lecteurs d'écran et traités comme s'ils faisaient partie de la page.

a) Les titles des cadres

Les Iframes qui proposent du contenu aux utilisateurs doivent avoir un title non vide, informatif et explicite. Il doit par ailleurs être unique dans le contexte de la page (pas deux fois le même sur la même page)

Le title ne pourra pas être générique et contenir un texte du type « vidéo » ou « publicité ».

Bon à savoir : JAWS lit en priorité le title du document inclus dans l'Iframe. Si celui-ci est absent, JAWS lira le title de l'Iframe. Dans l'idéal, pour éviter les erreurs de compréhension, on renseignera donc un title d'Iframe similaire ou proche du title du contenu source et on s'assurera que celui-ci contienne un title correct (en supposant avoir la main sur ce contenu).

Bon exemple d'utilisation des title dans des Iframe

```
<iframe

title="Specify the Language" width="560" height="315" src="https://www.youtube.com/embed/qyjDrUV_el8" frameborder="0" allowfullscreen>

</iframe>

<iframe

title="Video of touch screen for the blind in New York City taxis" style="width:640px;height:480px;margin:auto;text-align:center;" src="//www.youtube.com/embed/hM0x0k2Bv3Y" allowfullscreen>

</iframe>
```

b) La structure sémantique des cadres

Le contenu des cadres fait partie intégrante de la page et est donc perçu comme tel par le lecteur d'écran. Si la page contient 5 niveaux de titre et le cadre 3, le lecteur d'écran listera 8 niveaux de titres.

Dès lors, dans la mesure du possible, si le contenu de l'Iframe peut être modifié, on s'assurera que sa structure s'intègre dans la structure où il sera affiché. On évitera par exemple de trouver plusieurs <h1> sur la page.

c) Cacher des cadres qui ne présentent pas de contenus significatifs

Certains contenus n'apportent rien à l'utilisateur, sont considérés comme du contenu décoratif ou peuvent même compliquer la lecture par les lecteurs d'écran (ex : Google maps).

Dans ce cas, il est préférable de cacher l'Iframe grâce à l'attribut aria-hidden= « true »

Exemple d'utilisation d'aria pour cacher du contenu aux lecteurs d'écran

```
<iframe

 title="Intentionally blank" aria-hidden="true" src="http://bit.ly/2cfBoyE" width="120" height="50">

</iframe>
```

Tableaux

Les utilisateurs de lecteurs d'écran peuvent naviguer dans les tableaux de cellule en cellule. Si les tableaux sont correctement structurés, l'internaute entendra le lien entre les entêtes et les cellules correspondantes. Des moyens doivent également être prévus pour permettre une bonne compréhension des tableaux pour tous.

Une bonne sémantique pour les tableaux de données

Les tableaux de données doivent utiliser les balises html destinées aux tableaux <table>… Certains faux tableaux sont générés en CSS pour permettre une meilleure lisibilité notamment sur les smartphones.

Les utilisateurs de lecteurs d'écran ne comprendront pas les liens entre les cellules et les entêtes et ne pourront naviguer comme dans un tableau sans la bonne sémantique.

Les légendes des tableaux

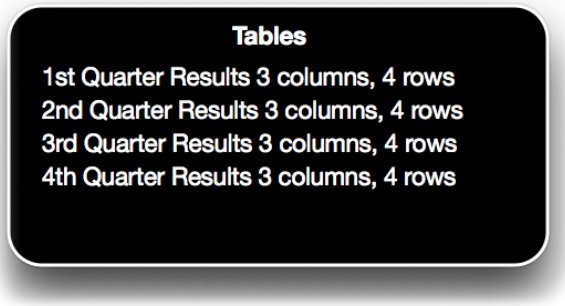
a) *Les tableaux de données devraient être techniquement associés à une légende ou à un nom.*

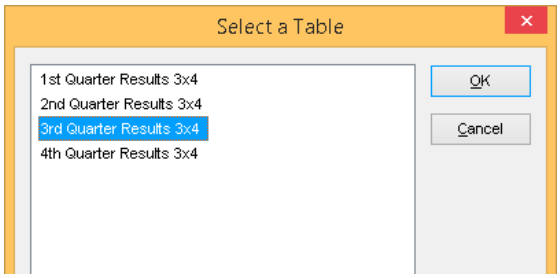
Les lecteurs d'écran lisent la légende ou le nom associé au tableau lorsque l'utilisateur navigue dans le tableau.

L'élément <caption> est la technique la plus directe à cet effet mais il est également possible d'utiliser aria-label ou aria-labelledby.

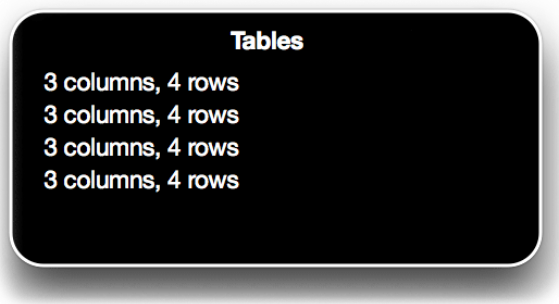
L'internaute peut, à l'aide de son lecteur d'écran, lister les tableaux présents sur la page. La légende du tableau sera également lue, si elle est présente. Le cas échéant, la liste ne contiendra qu'un aperçu des tableaux (nombre de lignes et colonnes) sans aucune autre indication.

Les deux images ci-dessous montrent la liste de 4 tableaux correctement légendés (VoiceOver et Jaws)





L'image ci-dessous montre les mêmes tableaux non légendés en VoiceOver



Bon exemple : un tableau légendé avec l'élément <caption>

```
<table>
 <caption>1st Quarter Results (EUR)</caption>
  <thead>
  <tr>
    <td> </td>
    <th scope="col">Goal</th>
    <th scope="col">Actual </th>
  </tr>
  </thead>
  <tbody>
  <tr>
    <th scope="row">January</th>
    <td>200</td>
    <td>150</td>
  </tr>
  <tr>
    <th scope="row">February</th>
    <td>200</td>
    <td>240</td>
   </tr>
   <tr>
     <th scope="row">March</th>
     <td>200</td>
     <td>370</td>
```

```
    </tr>
    </tbody>
</table>
```

Bon exemple : Une légende avec aria-label

```
<table aria-label="Second quarter Results">
```

Attention, le texte repris dans aria-label n'est pas visible que pour les utilisateurs de lecteurs d'écrans.

Bon exemple : Une légende cachée en CSS

```
<table>
    <caption><span class="visually-hidden">1st Quarter Results</span></caption>
```

Une classe en CSS est utilisée pour cacher la légende mais la rendre lisible par les lecteurs d'écrans. Attention, la class ne peut être placée directement sur le <caption> car NVDA ne lira alors pas le nombre correct de lignes.

Bon exemple : Une légende en aria-labelledby

```
<h3 id="tableCaption">Third Quarter Results</h3>

<table aria-labelledby="tableCaption">
```

Une variation de cette méthode consiste à placer le nom dans un <figcaption> et de l'associer à la table à l'aide de arie-labelledby

```
<figure>

<table aria-labelledby="figCaption">

<!—data table-->

</table>

<figcaption id="figCaption">Fourth Quarter Results</figcaption>

</figure>
```

Mauvais exemple : Un tableau avec une fausse légende

Ce tableau ne contient pas de légende mais commence par une ligne d'entête qui fusionne 3 cellules. Non seulement la légende ne sera pas correctement gérée mais la navigation dans le tableau sera confuse et difficile.

```
<table>
  <tr>
    <th colspan="3">Revenue Goals (EUR)</th>
  </tr>
  <tr>
    <td> </td>
    <th scope="col">Goal</th>
    <th scope="col">Actual</th>
```

```
  </tr>
  <tr>
    <th scope="row">January</th>
    <td>200</td>
    <td>150</td>
  </tr>
  <tr>
    <th scope="row">February</th>
    <td>200</td>
    <td>240</td>
   </tr>
   <tr>
     <th scope="row">March</th>
     <td>200</td>
     <td>370</td>
   </tr>
</table>
```

Mauvais exemple : Un titre de tableau non associé avec le tableau

```
<h3>Revenue Goals</h3>

<table>

<!—data table -->
```

Le titre <h3> ressemblera au titre de tableau mais, non associé techniquement, le lecteur d'écran ne pourra pas faire le lien.

b) La légende du tableau devrait décrire l'objet du tableau de manière unique explicite, significative et succincte.

Le but de cette règle est de permettre d'identifier rapidement et sans erreur le bon tableau.

Dans le même esprit, si la page contient plusieurs tableaux, chaque tableau devrait avoir un titre unique.

Les entêtes de tableaux de données

a) Les entêtes de tableaux de données doivent être balisés par <th>

La balise <th> permet de préciser à l'utilisateur de lecteur d'écran qu'il se trouve sur une cellule d'entête.

Il est recommandé de rendre le scope de l'entête explicite en ajoutant un attribut scope= « col » ou « row » ou « colgroup » ou « rowgroup » à la balise <th>.

b) *Les entêtes de tableaux de données doivent décrire spécifiquement la catégorie du contenu des cellules*

Le texte de l'entête ne peut pas être vague et informer clairement l'utilisateur sur le contenu des cellules. Par exemple, ne pas écrire « colonne 2 » et décrire explicitement l'alternative d'un bouton ou lien.

La règle est d'être court et clair.

Les associations entre l'entête et les cellules des tableaux de données

a) *Les cellules de données doivent être associées avec leur cellule d'entête correspondante*

L'attribut scope crée une association explicite entre la cellule d'entête et la cellule de données correspondante. Les options sont :

- Scope= « col » pour les entêtes de colonnes
- Scope= « row » pour les entêtes de lignes

Bon exemple : Un tableau simple avec des entêtes de lignes et de colonnes

Dans le tableau ci-dessous, la première ligne contient les entêtes relatifs aux cellules de chaque colonne et la première colonne contient les entêtes relatifs aux cellules de chaque ligne.

Greensprings Running Club Personal Bests

Name	1 mile	5 km	10 km
Mary	8:32	28:04	1:01:16
Betsy	7:43	26:47	55:38
Matt	7:55	27:29	57:04
Todd	7:01	24:21	50:35

```
<table class="data">
<caption><strong>Greensprings Running Club Personal Bests</strong></caption>
<thead>
<tr>
<th scope="col">Name</th>
<th scope="col">1 mile</th>
<th scope="col">5 km</th>
<th scope="col">10 km</th>
</tr>
</thead>
<tbody>
```

```
<tr>
<th scope="row">Mary</th>
<td>8:32</td>
<td>28:04</td>
<td>1:01:16</td>
</tr>
<tr>
<th scope="row">Betsy</th>
<td>7:43</td>
<td>26:47</td>
<td>55:38</td>
</tr>
<tr>
<th scope="row">Matt</th>
<td>7:55</td>
<td>27:29</td>
<td>57:04</td>
</tr>
<tr>
<th scope="row">Todd</th>
<td>7:01</td>
<td>24:21</td>
<td>50:35</td>
</tr>
</tbody>
</table>
```

b) *Les cellules de données doivent être associées avec leurs cellules d'entêtes correspondantes dans entêtes de tableaux groupés*

Les utilisateurs de lecteurs d'écran et tous les utilisateurs en général peuvent éprouver des difficultés à comprendre et à naviguer sur ce type de tableaux avec des entêtes groupés. Dans la mesure du possible, il est conseillé de structurer les données dans des tableaux simples (une ligne ou une colonne d'entêtes, sans fusion de cellules).

Le meilleur moyen d'indiquer un regroupement de cellules d'entêtes aux lecteurs d'écran est l'utilisation de scope= « rowgroup » ou scope= « colgroup ».

Bon exemple : scope= « colgroup » appliqué à des cellules fusionnées d'entêtes de colonnes

Table with colgroup

	Females			Males		
	Mary	Betsy	Joanne	Matt	Todd	Jake
1 mile	8:32	7:43	9:51	7:55	7:01	7:51
5 km	28:04	26:47	38:15	27:27	24:21	24:31
10 km	1:01:16	55:38	1:56:01	57:04	50:35	50:45

```
<table class="data complex">
  <caption>
    Table with colgroup
  </caption>
  <thead>
    <tr>
      <td rowspan="2">&amp;nbsp;</td>
      <th colspan="3" scope="colgroup">Females</th>
      <th colspan="3" scope="colgroup">Males</th>
    </tr>
    <tr>
      <th scope="col">Mary</th>
      <th scope="col">Betsy</th>
      <th scope="col">Joanne</th>
      <th scope="col">Matt</th>
      <th scope="col">Todd</th>
      <th scope="col">Jake</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <th scope="row">1 mile</th>
      <td>8:32</td>
      <td>7:43</td>
```

```
    <td>9:51</td>
    <td>7:55</td>
    <td>7:01</td>
    <td>7:51</td>
  </tr>
  <tr>
    <th scope="row">5 km</th>
    <td>28:04</td>
    <td>26:47</td>
    <td>38:15</td>
    <td>27:27</td>
    <td>24:21</td>
    <td>24:31</td>
  </tr>
  <tr>
    <th scope="row">10 km</th>
    <td>1:01:16</td>
    <td>55:38</td>
    <td>1:56:01</td>
    <td>57:04</td>
    <td>50:35</td>
    <td>50:45</td>
  </tr>
 </tbody>
</table>
```

Bon exemple : scope= « rowgroup » appliqué à des cellules fusionnées d'entêtes de lignes

Cet exemple montre le même tableau présenté différemment.

Les lecteurs d'écran supportent plus difficilement le scope= « rowgroup ». Donc, dans la mesure du possible, il est recommandé de structurer le tableau complexe pour un usage scope= « colgroup ».

Table with rowgroup

		1 mile	5 km	10 km
Females	Mary	8:32	28:04	1:01:16
	Betsy	7:43	26:47	55:38
	Joanne	9:51	38:15	1:56:01
Males	Matt	7:55	27:29	57:04
	Todd	7:01	24:21	50:35
	Jake	7:51	24:31	50:45

```
<table class="data complex">
  <caption>
    Table with rowgroup
  </caption>
  <thead>
    <tr>
      <td colspan="2">&amp;nbsp;</td>
      <th scope="col">1 mile</th>
      <th scope="col">5 km</th>
      <th scope="col">10 km</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <th rowspan="3" scope="rowgroup">Females</th>
      <th scope="row">Mary</th>
      <td>8:32</td>
      <td>28:04</td>
      <td>1:01:16</td>
    </tr>
    <tr>
      <th scope="row">Betsy</th>
      <td>7:43</td>
```

```
      <td>26:47</td>
      <td>55:38</td>
    </tr>
    <tr>
      <th scope="row">Joanne</th>
      <td>9:51</td>
      <td>38:15</td>
      <td>1:56:01</td>
    </tr>
    <tr>
      <th rowspan="3" scope="rowgroup">Males</th>
      <th scope="row">Matt</th>
      <td>7:55</td>
      <td>27:29</td>
      <td>57:04</td>
    </tr>
    <tr>
      <th scope="row">Todd</th>
      <td>7:01</td>
      <td>24:21</td>
      <td>50:35</td>
    </tr>
    <tr>
      <th scope="row">Jake</th>
      <td>7:51</td>
      <td>24:31</td>
      <td>50:45</td>
    </tr>
  </tbody>
</table>
```

c) Dans les tableaux complexes, les cellules de données qui ne peuvent être associées avec leurs cellules d'entêtes correspondantes avec <th> et scope doivent être désignées avec headers + id

Les tableaux complexes, où des cellules de données sont fusionnées ou qui contiennent 2 niveaux d'entêtes nécessitent la mise en place d'une technique différente que le marquage par <th> et scope pour faire le lien entre les cellules liées.

La méthode consiste à lier les cellules avec headers et id.

Attention, ce type de tableaux peut être difficile à comprendre et la navigation avec un lecteur d'écran y restera plus difficile même si les techniques d'accessibilité sont mises en place. Il est donc toujours recommandé de simplifier au maximum la structure des tableaux.

Certains lecteurs d'écran, spécialement sur mobile, ne gèrent pas les marquages complexes. Les anciennes versions de VoiceOver sur OS X jusque Mac OS X 10.10.2 présentent des erreurs de lectures sur des tableaux complexes qui utilisent les associations headers + id.

Bon exemple : Un tableau complexe avec Headers & ID

L'exemple ci-dessous décrit un tableau complexe qui répond aux standards d'accessibilité mais il n'en reste pas moins difficile à comprendre et la navigation via un lecteur d'écran est plus compliquée voire impossible avec certaines versions plus anciennes de lecteurs d'écran.

Note importante : Les ID's ne peuvent commencer par des chiffres mais uniquement par des lettres.

New Employee Orientation Schedule

Date	Schedule		Location	Topics
	Start	End		
Monday, June 1	9:00 a.m.	10:30 a.m.	RH 001	Introduction to Company: Vision and Mission
	10:30 a.m.	12:00 p.m.	RH 001	HR Policies Review
	Lunch from 12:00 p.m. to 1:00 p.m.			
	1:00 p.m.	2:30 p.m.	RH 001	Overview of Benefits
	3:00 p.m.	4:30 p.m.	RH 005	Health and Safety Procedures

```
<table class="complexexample">
 <caption>New Employee Orientation Schedule</caption>
 <tbody>
  <tr>
   <th rowspan="2" id="date">Date</th>
   <th colspan="2" id="schedule">Schedule</th>
   <th rowspan="2" id="location">Location</th>
   <th colspan="2" rowspan="2" id="topics1">Topics</th>
```

```
</tr>
<tr>
  <th id="start">Start</th>
  <th id="end">End</th>
</tr>
<tr>
  <th id="monday" rowspan="5">Monday, June 1</th>
  <td headers="schedule start monday">9:00 a.m.</td>
  <td headers="schedule end monday">10:30 a.m.</td>
  <td headers="location monday">RH 001</td>
  <td headers="topics1 monday">
    Introduction to Company: Vision and Mission</td>
</tr>
<tr>
  <td headers="schedule start monday">10:30 a.m.</td>
  <td headers="schedule end monday">12:00 p.m.</td>
  <td headers="location monday">RH 001</td>
  <td headers="topics1 monday">HR Policies Review</td>
</tr>
<tr>
  <td headers="schedule monday" colspan="5">
    <strong><em>
      Lunch from 12:00 p.m. to 1:00 p.m.
    </em></strong>
  </td>
</tr>
<tr>
  <td headers="schedule start monday">1:00 p.m.</td>
  <td headers="schedule end monday">2:30 p.m.</td>
  <td headers="location monday">RH 001</td>
  <td headers="topics1 monday">Overview of Benefits</td>
</tr>
```

```
  <tr>
    <td headers="schedule start monday">3:00 p.m.</td>
    <td headers="schedule end monday">4:30 p.m.</td>
    <td headers="location monday">RH 005</td>
    <td headers="topics1 monday">
      Health and Safety Procedures
    </td>
  </tr>
 </tbody>
</table>
```

Bien que cette structure soit fastidieuse à créer en html « à la main », cette approche est techniquement facile à programmer côté serveur avec des langages tels que PHP, .net, JSP, Python… pour créer des tableaux affichent les exports de bases de données.

d) Les tableaux imbriqués ou splittés

Dans les tableaux qui se présentent comme un tableau unique mais qui contiennent plusieurs parties de tableaux imbriqués, les techniques d'association d'entêtes avec les cellules vues plus haut ne fonctionnent pas (scope et headers + id) car il n'est pas possible d'associer des cellules de tableaux différents.

La seule solution consisterait à utiliser aria-labelledby mais le travail sera considérable juste pour compenser un tableau mal prévu au départ.

Mauvais exemple : Des tableaux imbriqués

Greensprings Running Club Personal Bests

		1 mile	5km	10km
Females	Mary	8:32	28:04	1:01:16
	Betsy	7:43	26:47	55:38
Males	Matt	7:55	27:29	57:04
	Todd	7:01	24:21	50:35

```
<table class="data">
<caption>
Greensprings Running Club Personal Bests
</caption>
<tr>
<td> </td>
<td style="padding:0">
```

```
<table style="margin:0;padding:0">
<tr>
<td style="width:60px;"> </td>
<th style="width:60px">1 mile</th>
<th style="width:60px">5km</th>
<th style="width:60px">10km</th>
</tr>
</table></td>
</tr>
<tr>
<th>Females</th>
<td style="padding:0">
<table style="margin:0;padding:0">
<tr>
<th style="width:60px;">Mary</th>
<td style="width:60px">8:32</td>
<td style="width:60px">28:04</td>
<td style="width:60px">1:01:16</td>
</tr>
<tr>
<th style="width:60px;">Betsy</th>
<td>7:43</td>
<td>26:47</td>
<td>55:38</td>
</tr>
</table>
</td>
</tr>
<tr>
<th>Males</th>
<td style="padding:0">
<table style="margin:0;padding:0">
```

```
<tr>
<th style="width:60px;">Matt</th>
<td style="width:60px">7:55</td>
<td style="width:60px">27:29</td>
<td style="width:60px">57:04</td>
</tr>
<tr>
<th style="width:60px;">Todd</th>
<td>7:01</td>
<td>24:21</td>
<td>50:35</td>
</tr>
</table></td>
</tr>
</table>
```

Les résumés de tableaux (summary)

a) *Un résumé du tableau devrait être fourni pour les tableaux de données*

Un résumé n'est pas obligatoire. La structure du tableau et les titres des entêtes devraient être suffisants pour comprendre les données du tableau.

Le cas échéant, si le tableau demande un complément d'information pour sa bonne compréhension, un résumé peut en être donné.

Le résumé est destiné à aider l'utilisateur à comprendre le tableau. En aucun cas il ne doit servir à préciser des mots clés pour le référencement.

Le contenu devra être concis et explicite. D'un maximum de deux phrases, il pourra être un peu plus long pour décrire des tableaux complexes. Il existe 4 manières de proposer un résumé :

1. Le résumé est écrit dans un paragraphe qui précède ou suit le tableau. Le paragraphe associé au tableau avec aria-describedby.
2. Un résumé, s'il est très court peut être écrit dans le <caption> du tableau.
3. Le tableau est dans une balise <figure> et son titre et son résumé sont écrits dans le <figcaption>.
4. Uniquement pour le HTML antérieur à HTML5, le résumé est dans l'attribut summary. Déprécié donc non recommandé.

Bon exemple : un résumé dans un paragraphe associé avec aria-describedby (méthode 1)

Un texte écrit dans un paragraphe est lisible par tout le monde. L'association explicite avec aria-describedby permet aux lecteurs d'écrans de faire le lien entre le paragraphe et le tableau.

```
<p id="table-description">This table lists the members of the
```

Greensprings Running Club and their personal best times in various race distances.

The first column lists the runners and the first row lists the race distances.</p>

```
<table aria-describedby="table-description">
  <caption>Greensprings Running Club Personal Bests</caption>
  <tr>
    <th scope="col">Name</th>
    <th scope="col">1 mile</th>
    <th scope="col">5 km</th>
    <th scope="col">10 km</th>
    </tr>
  <tr>
    <th scope="row">Mary</th>
    <td>8:32</td>
    <td>28:04</td>
    <td>1:01:16</td>
    </tr>
  <tr>
    …..
</table>
```

Il est possible de cacher visuellement le résumé en appliquant une classe en CSS qui utilise la méthode clip décrite dans les autres chapitres. Cette méthode est parfaitement accessible aux utilisateurs de lecteurs d'écran mais le contenu ainsi caché pourrait pourtant être utile à d'autres internautes qui éprouvent des difficultés à comprendre le tableau.

```
<p id="table-description" class="visually-hidden">This table lists the members of the
Greensprings Running Club and their personal best times in various race distances.
The first column lists the runners and the first row lists the race distances.</p>
```

Bon exemple : un résumé court dans le <caption> du tableau (méthode 2)

```
<table>
    <caption>
      Greensprings Running Club Personal Bests<br>
      (The first column lists the runners and the
      first row lists the race distances)
    </caption>
  <tr>
```

```
  <th scope="col">Name</th>
  <th scope="col">1 mile</th>
  <th scope="col">5 km</th>
  <th scope="col">10 km</th>
  </tr>
 …….
</table>
```

Bon exemple : Un tableau dans la balise <figure> dont le résumé est placé dans la balise <figcaption> (méthode 3)

Bon à savoir : La lecture du contenu d'une balise <figure> par un lecteur d'écran est plus longue car elle peut nécessiter plus d'interactions et le mot <figure> est plus générique donc l'utilisateur ne comprendra pas immédiatement le type d'élément contenu.

Mais la technique est parfaitement accessible.

```
<figure>
  <figcaption id="table_figcaption">
    Greensprings Running Club Personal Bests<br>
    (The first column lists the runners and the first row lists the race distances)
  </figcaption>
<table aria-labelledby="table_figcaption">
 <tr>
  <th scope="col">Name</th>
  <th scope="col">1 mile</th>
  <th scope="col">5 km</th>
  <th scope="col">10 km</th>
  </tr>
 <tr>
  <th scope="row">Mary</th>
  <td>8:32</td>
  <td>28:04</td>
  <td>1:01:16</td>
  </tr>
 <tr>
  <th scope="row">Betsy</th>
  <td>7:43</td>
```

```
    <td>26:47</td>
    <td>55:38</td>
    </tr>
  <tr>
    <th scope="row">Matt</th>
    <td>7:55</td>
    <td>27:29</td>
    <td>57:04</td>
    </tr>
  <tr>
    <th scope="row">Todd</th>
    <td>7:01</td>
    <td>24:21</td>
    <td>50:35</td>
    </tr>
</table>
<figure>
```

Bon exemple : Un tableau dans la balise <figure> dont au moins le résumé est placé dans le même <figure> (méthode 3)

```
<figure>
  <figcaption>
    Greensprings Running Club Personal Bests
  </figcaption>
  <p>(The first column lists the runners and the  first row lists the race distances) </p>
<table>
 …
</table>
</figure>
```

Les tableaux de mise en forme

a) Les tableaux ne devraient pas être utilisés à des fins de mise en page (layout).

Les tableaux ne devraient être utilisés que pour présenter des données. Sémantiquement, ils ont été créés à cet effet. Les CSS sont plus indiqués pour la mise en page et sont par ailleurs plus robustes et répondent mieux au web moderne.

La navigation dans un tableau à l'aide d'un lecteur d'écran est plus longue et l'utilisateur entend qu'il est dans un tableau où il s'attend à trouver des données présentées dans des cellules avec des entêtes…

Le cas échéant, si un tableau devait être utilisé, le role= « presentation » devrait y être spécifié pour permettre à l'utilisateur de lecteur d'écran d'entendre son contenu en tant que texte linéaire.

b) Les tableaux de mise en forme ne peuvent pas contenir d'éléments sémantiques liés aux tableaux.

Si le tableau contient des éléments de balisage sémantique spécifiques aux tableaux, le lecteur d'écran les lira comme tels laissant croire à l'utilisateur qu'il est occupé à lire un tableau de données.

En d'autres termes, le tableau ne pourra pas contenir les éléments suivants :

- L'élément <caption>
- L'attribut <summary>
- L'élément <th>
- L'attribut scope
- Les attributs headers + id

P. Composants dynamiques

Des composants dynamiques peuvent être créés de manière accessible en JavaScript. La documentation relative à chaque composant est disponible en ligne sur le site du W3C-WAI

Ce document ne reprendra donc pas l'ensemble des ressources régulièrement mises à jour sur le site officiel mais donnera quelques compléments d'information et quelques exemples.

Concepts ARIA

ARIA (Accessible Rich Internet Applications) a été créé spécifiquement pour l'accessibilité du web et en particulier pour les technologies d'assistance. Il s'agit d'éléments qui permettent de compenser les restrictions en HTML5 et / ou de corriger des sites où il est difficile de revoir la structure pour toute une série de raisons.

Il faut avoir à l'esprit les éléments suivants avant d'utiliser ARIA :

- Ne jamais utiliser ARIA sauf en cas d'obligation.
- Toujours utiliser ARIA lorsque c'est nécessaire.
- Par défaut, sans connaissances, on utilise mal ARIA.

Autrement dit, ARIA viendra toujours en dernier recours et il faudra l'utiliser en connaissance de cause.

Les composants dynamiques en JavaScript utilisent souvent ARIA pour les raisons évoquées ci-dessus. Chaque composant (widget) doit respecter des critères d'accessibilité pour répondre aux besoins des internautes, utilisateurs ou non de technologies d'assistance (navigation à la souris uniquement, au clavier uniquement, via les technologies d'assistance…).

a) Que fait ARIA ?

ARIA permet de communiquer les informations suivantes aux lecteurs d'écran :

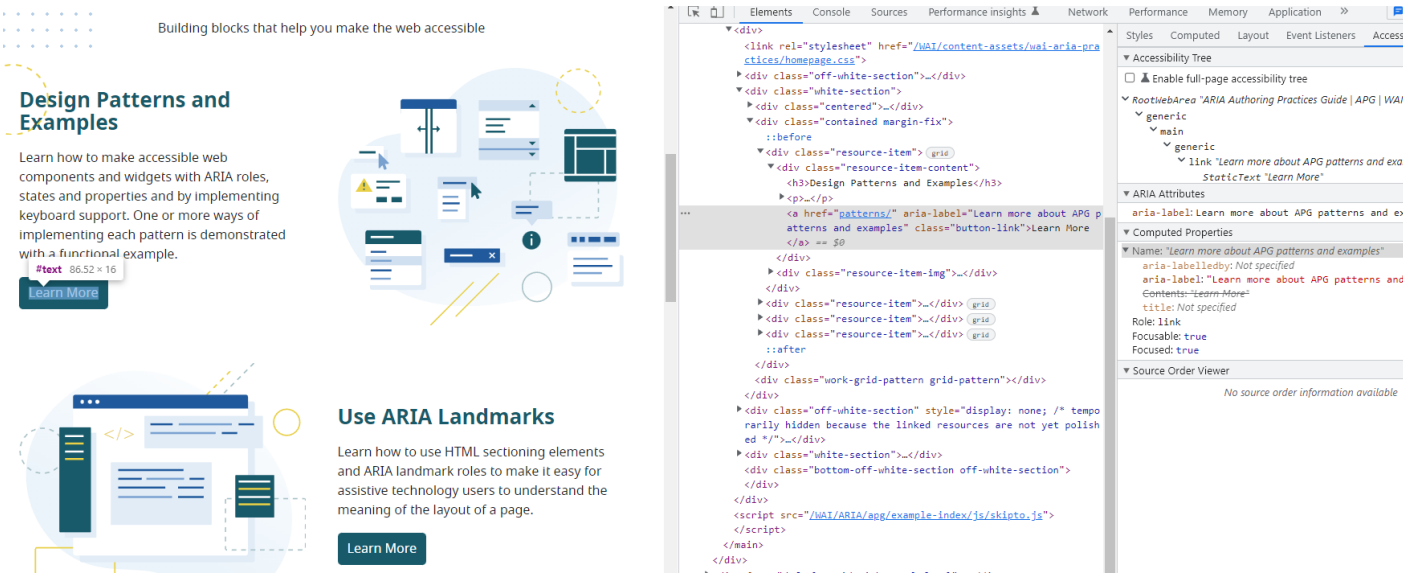
- Les labels (labels) : Le nom des éléments. Exemple : aria-label.
- Les rôles (roles) : L'objet des éléments. Exemple : role= « navigation »
- Les états (states) des éléments dynamiques : Leur statut. Exemple : aria-selected= « true »
- Les propriétés (properties). Exemple : aria-haspopup= « true »
- Les relations (relationships) entre éléments. Exemple : aria-controls
- Les annonces (live announcements) en temps réel. Les live regions qui annoncent un changement sur la page.

b) Le nom accessible (Name)

Le nom accessible est le nom de l'élément tel qu'il sera généré par l'accessibility tree, écrit dans le DOM, communiqué au lecteur d'écran et lu à l'utilisateur.

Dans l'illustration ci-dessous, l'onglet « Accessibility » de l'inspecteur de code montre le nom accessible « Learn more about APG patterns examples » du lien « Learn More ». Le nom accessible a été généré sur base de l'attribut aria-label du lien.

L'illustration montre également le role : link (lien), l'état et la valeur de l'élément (Focusable : true, Focused : true).



Calcul du nom accessible (algorithme)

De manière simplifiée, voici l'ordre dans lequel le nom accessible est calculé :

1. Aria-labelledby. Le texte de cet attribut, s'il est présent, écrase tous les autres textes de labellisation de l'élément.
2. Aria-label. Sans aria-labelledby, le texte du aria-label écrase les autres textes. Dans l'illustration précédente, le contenu du lien présent entre les balises <a> et était « Learn More ». Sans le texte de l'attribut aria-label= « Learn more about APG patterns and examples », le nom accessible aurait été « Learn More » (non explicite hors contexte). Il faut bien noter que l'aria-label est invisible à l'écran et uniquement disponible pour les utilisateurs de lecteurs d'écran.
3. Le texte de l'élément natif HTML ou son label ou son alternative textuelle. Sans les 2 attributs ci-dessus, le texte natif constitue le nom accessible. Il peut s'agir du texte entre les balises d'ouverture et de fermeture d'un bouton (<button>), du <label> d'un champ de formulaire correctement associé, d'une alternative d'image (alt), du texte compris entre les balises de lien <a> et (alt et title d'image compris)…
4. L'attribut title. Si aucun autre élément ci-dessous ne contient de texte, le title servira en dernier lieu au calcul du nom accessible. Attention, le title n'est pas considéré comme technique accessible car son contenu n'est visible qu'au survol de souris le rendant donc invisible aux utilisateurs de clavier.

Même si le texte de l'élément natif n'apparait qu'en troisième position, il s'agit de la méthode à privilégier en premier lieu.

Aria-labelledby

Le texte du aria-labelledby est en général visible sur la page par tous les utilisateurs voyants (à l'inverse de aria-label). L'objet est d'associer techniquement un élément à ce texte (label) par un id pour qu'il soit lu par le lecteur d'écran au passage sur l'élément. Dans certains rares cas, ce texte pourra être rendu invisible à l'écran désavantageant alors les personnes voyantes.

L'attribut aria-labelledby renseigne l'id de l'élément qui contient le texte. Il ne contient pas le texte (à l'inverse de aria-label).

Attention, aria-labelledby remplace le label existant, il n'y ajoute pas de texte complémentaire. Même si, dans certaines configurations, le lecteur d'écran lire le label d'origine et le aria-labelledby. Mais ce cas de figure ne doit pas être pris en compte.

Exemple de aria-labelledby

Dans cet exemple, la div qui englobe le contenu du pop-up a le nom de son <h1> identifié par l'id « h1 » : « Confirm your selection »

<h1 id="h1">Confirm your selection</h1>

Une des forces de cet attribut est de pouvoir se référer à plusieurs éléments. Ce qui peut être très intéressant pour des formulaires où des champs doivent reprendre plusieurs informations différentes pour aider l'utilisateur à les compléter.

Exemple de aria-labelledby avec plusieurs ID's

 ... ...

<input type="text" aria-labelledby="males frank ranking">

Aria-label

L'attribut aria-label contient le texte du label lui-même et n'est pas visible à l'écran. Seul l'utilisateur d'un lecteur d'écran pourra en prendre connaissance.

Exemple de aria-label

L'élément de navigation ci-dessous aura comme nom le contenu de l'attribut aria-label.

<nav aria-label="Clothing for Girls">

Le texte du aria-label remplace le texte existant. Il n'est pas destiné à donner une information complémentaire puisqu'elle n'est visible que par les lecteurs d'écran.

Exemple de aria-label qui remplace le texte natif d'un lien

Dans l'exemple ci-dessous, les utilisateurs de lecteurs d'écrans n'entendent pas la même information que celle visible sur la page.

W3C

Exemple de aria-label sur des régions Landmark

Si le site ne contient qu'une navigation principale, il n'est pas nécessaire de lui donner un nom. Le lecteur d'écran lira « Navigation ».

Par contre, si la page contient plusieurs navigations, il peut être utile de les nommer pour permettre à l'utilisateur de naviguer plus facilement en différenciant les différents menus de navigation.

<nav role="navigation" aria-label="Product Categories">

Exemple de aria-label sur un moteur de recherche

Il existe des cas spécifiques où l'élément ne doit pas nécessairement posséder de label. C'est le cas d'un formulaire de recherche où la proximité du bouton « Search » indique visuellement à l'objet du champ à l'utilisateur voyant.

Malheureusement, la personne aveugle ne déduira l'objet du champ qu'en continuant à naviguer plus loin. Pour elle, ce manque de label peut s'avérer problématique.

Dans ce cas, aria-label est un moyen de lui rendre ce champ explicite tout en n'ajoutant pas de label supplémentaire à l'écran.

```
<form action="#" role="search">
  <input aria-label="Site Search" name="search" type="search">
  <input type="submit" value="Search">
</form>
```

c) Role

Chaque élément html possède un role qui comporte des propriétés et des méthodes pour fournir des informations à l'utilisateur. Les technologies d'assistance déduisent du rôle de l'élément quels moyens d'interaction ils doivent proposer à l'utilisateur.

Par exemple, la balise <img> indique le role d' « image » au lecteur d'écran qui indiquera à l'utilisateur qu'il se trouve sur une « image » ou « graphique » en fonction des lecteurs et il lira le texte alternatif contenu dans l'attribut alt. Si l'attribut alt est manquant, le lecteur d'écran lira le chemin et / ou le nom complet de l'image. Le fonctionnement sera différent sur un <p> qui indique le role « paragraph »…

Différents roles

Il existe différents roles dont certains ont déjà été abordés dans cette documentation. Le site du W3C-ARIA explique en détail chacun d'entre eux. Nous ne les reprendrons donc pas en détail. Ces rôles sont regroupés dans des catégories : Landmark, Widget, Pseudo, Document, Application, Presentation, Math, Definition, Note, Directory et Abstract. Nous renvoyons le lecteur vers la documentation officielle pour une description détaillée.

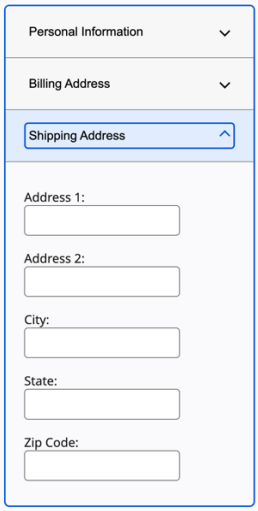
Précisions juste que chaque élément de la page peut recevoir un role html (pseudo html roles). Par exemple, le paragraphe <p role= « heading » aria-level= « 1 »> sera considéré comme un titre de niveau 1 <h1>.

Par ailleurs, tous les composants dynamiques (widgets) ont un role spécifique qui répondent à des critères à tenir en compte pour interagir avec les technologies d'assistance. Les 30 roles widgets sont décrits en détail dans les ressources en ligne du W3C-ARIA sous le vocable de Design Patterns

Pour chaque composant, le site donne les interactions obligatoires avec le clavier et les technologies d'assistance, des exemples de composants, du code source (html, css et JavaScript), les rôles (roles), états (states) et propriétés (Properties)…

Exemple de composant dynamique : Expand / Collapse (accordéon)

Le pattern est constitué d'un bouton bascule qui découvre un contenu caché (collapsed) ou qui cache un contenu déroulé à l'écran (expanded). Ce composant est souvent utilisé dans des formulaires ou les FAQ pour ne pas afficher trop de contenu à la fois sur une page.



Au clavier, la touche ENTER ou la barre d'espace doivent ouvrir ou fermer l'élément (focusable).

Le lecteur d'écran doit entendre qu'il s'agit d'un bouton, lire le texte du bouton et dire si le contenu est plié ou déplié (Collapse / Expand).

Si les ressources techniques sont bien respectées, toutes les combinaisons de lecteurs d'écran + navigateur peuvent interagir avec ce composant.

Informations sur le design pattern accordion

Documentation technique et exemple du composant accordion

d) Valeur (value)

Les propriétés et les états des composants doivent être communiqués aux technologies d'assistance mais également leurs changements.

Dans certains cas, ces attributs se réfèrent aux ID's d'autres éléments (ex : aria-labelledby= unID »). Dans d'autres cas, les options sont « true » ou « false » (ex : aria-hidden= « true »). Et dans d'autres cas de figure les attributs sont limités à des options prédéfinies (ex : aria-orentation= « vertical » ou « horizontal »).

Pour les composants dynamiques, la valeur doit être modifiée en JavaScript sur base des interactions avec l'utilisateur ou de certaines circonstances (ex : aria-expanded= « true » ou « false »).

L'ensemble des valeurs, les états et propriétés sont décrites sur le site du W3C-ARIA

e) Description (aria-describedby)

L'attribut aria-describedby est utilisé pour communiquer une information complémentaire à un élément. À l'inverse de aria-label et aria-labelledby, Aria-describedby n'entre pas en ligne de compte pour le calcul du nom accessible. Autrement dit, cet attribut ne peut être utilisé pour donner un nom, un label ou un titre à un élément. Si un élément ne nécessite qu'un nom accessible dans description complémentaire, il ne faut pas utiliser l'attribut aria-describedby.

Si l'élément comporte un nom et une description, les lecteurs d'écran lisent d'abord le nom et ensuite la description.

Si le texte du aria-describedby est critique (important), il doit être visible pour les personnes voyantes et doit être disponible en permanence dans le contexte du document de manière lisible par les lecteurs d'écran.

Dans l'exemple ci-dessous, une information complémentaire est communiquée via aria-describedby


```
<label for="newPassword">Choose a new password:</label>

<input type="password" id="newPassword" aria-describedby="pwdInfo">

<span id="pwdInfo">Minimum 8 characters, with both letters and numerals</span>
```

f) Live Regions

Les attributs aria-live annoncent aux lecteurs d'écrans les modifications du contenu des pages indépendamment de ce que l'utilisateur y fait au moment du changement. Les modifications peuvent être dues à l'utilisateur, basées sur un timer, sur le résultat d'un processus du serveur… Elles peuvent ou non être visibles à l'écran et ne sont habituellement pas accompagnées d'un changement de focus.

Techniquement, l'idée est de créer un container vide qui attend une injection en JavaScript. Dès que le contenu est injecté dans cette région, l'information est poussée au lecteur d'écran qui annonce le changement.

Bon exemple : Une annonce via aria live

Le conteneur est d'abord désigné comme région. Il doit être vide au chargement de la page ou lorsqu'il est ajouté au DOM (pas de texte entre les balises d'ouverture et de fermeture). Les annonces ne se font qu'au changement dans la live region. Aussi, si du texte est déjà présent avant le changement, il ne sera pas lu par le lecteur d'écran.

```
<div aria-live="polite"></div>
```

Au trigger, le contenu est injecté dans la div et le lecteur d'écran dit « Hello, screen reader user ! »

```
<div aria-live="polite">Hello, screen reader user!</div>
```

Assertive / polite

L'annonce aria-live peut être faite de deux manières : assertive et polite.

L'utilisateur n'a pas la possibilité de ré-écouter les annonces ou de les mettre en pause. Aussi, elles devraient être les plus brèves possibles pour éviter que l'internaute ne les coupe avec le risque de perdre une information importante.

Assertive : aria-live= « assertive »

- Le lecteur d'écran est interrompu dans sa lecture en cours et l'annonce est lue
- Après la lecture du message, la lecture ne reprend pas où le lecteur a été interrompu. L'utilisateur peut relire le contenu qu'il était occupé à écouter ou passer à un autre contenu.
- Les annonces ne sont pas mises en queue. Si plusieurs annonces se suivent, la première sera coupée par la suivante sans être relue.

Polite : aria-live= « polite »

- L'annonce est mise en queue dans l'attente que le lecteur finisse de lire le contenu en cours
- Sauf intervention de l'utilisateur, les annonces successives seront lues les unes après les autres dans leur ordre d'arrivée.

Les attributs des régions aria-live

Aria-atomic

- Aria-atomic= « false » (par défaut) génère la lecture uniquement du contenu qui a été changé (par exemple un gros titre dans un fil d'actualité).
- Aria-atomic= « true » relit la région entière où le changement est opéré pour donner plus de contexte à l'utilisateur.

Aria-relevant

- Aria-relevant= « all » : Tous les changements sont annoncés. À utiliser avec parcimonie pour ne pas nuire à l'ergonomie.
- Aria-relevant= « additions » : Seuls les ajouts de nœuds dans la région sont annoncés.
- Aria-relevant= « removals » : Seules les suppressions de nœuds dans la région sont annoncées. À n'utiliser que si le changement est important pour garder un confort de lecture.
- Aria-relevant= « text » : Seuls les changements de contenu textuel (y compris les textes alternatifs des alt) sont annoncés.

Autres principaux type de régions aria-live

Role= « alert »

Le role= « alert » est un type spécifique de live region assertive à utiliser pour annoncer une information importante aux lecteurs d'écran. Le fonctionnement est identique à aria-live= « assertive » avec l'ajout pour certains lecteurs de l'annonce « alerte ».

L'exemple ci-dessous montre du code à utiliser pour un message d'alerte qui avertit l'utilisateur sur le changement de ses préférences.

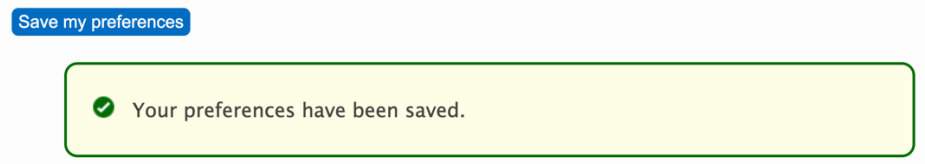
Le code HTML initial prévoit un conteneur vide qui accueillera le message d'alerte :

```
<form id="successForm" method="post" action="javascript:void(0)">
  <p><button>Save my preferences</button></p>
  <div class="msg" role="alert">
     <span class="msgTxt"></span>
  </div>
</form>
```

Le code final après injection du texte en JavaScript :

```
<form id="successForm" method="post" action="javascript:void(0)">
   <p><button>Save my preferences</button></p>
   <div class="msg" role="alert">
      <span class="msgTxt">Your preferences have been saved.</span>
   </div>
```

</form>



D'autres types de Live Region Roles sont décrits dans les ressources du W3C

g) L'accessibilité du clavier

Un site Internet doit pouvoir être complètement utilisé sans la souris, uniquement avec les touches du clavier. Un bon nombre d'utilisateurs en situation de handicap et de technologies d'assistance ne sont pas capables d'utiliser ou de fonctionner avec la souris.

Le premier test à réaliser sur un site Internet est de surfer uniquement au clavier avec la touche de tabulation, la touche ENTER, la barre d'espace et les flèches de navigation. Toutes les zones interactives sont-elles atteignables ? La prise de focus est-elle visible ?

Voici quelques recommandations à respecter.

Respect des Design patterns WAI-ARIA

Chaque design pattern des composants dynamiques et widgets liste les comportements au clavier à prévoir lors des développements.

Tab jusqu'au composant, flèches de navigation dans le composant

La tabulation doit permettre d'accéder au composant où on naviguera principalement avec les flèches de navigation (même si d'autres touches sont possibles en fonction des composants).

Une prise de focus toujours visible : Ne pas désactiver le outline

Par défaut, le outline des éléments focusables est suffisant. Il est toutefois possible de le renforcer en CSS aussi bien pour l'effet de survol, l'élément actif et la prise de focus.

Exemple de renforcement du outline en CSS :

```
a:focus, a:hover, a:active {
    background-color: #fdf6e7;
    outline: 1px solid #8cc63f;
}
```

Attention, comme dans l'exemple ci-dessous, beaucoup de reset CSS désactivent le outline. À vérifier avant utilisation !

```
a:focus {
    outline: 0; /* or outline: none; */
}
```

Un ordre de tabulation logique et cohérent entre le DOM et la présentation à l'écran

La tabulation doit être la plus logique possible et suivre l'ordre d'apparition dans le DOM sans changer l'ordre en CSS.

L'utilisateur s'attend à commencer en haut à gauche, puis de gauche à droite… jusqu'en bas à droite.

Éviter les pièges au clavier

L'utilisateur ne peut se trouver en situation de blocage dans un composant sans pouvoir en sortir.

Utiliser tabindex correctement

Tabindex= « 0 » pour rendre un élément tabulable et focusable

Un élément comme un <p>, <hx>, … ne nécessite en principe pas d'être atteignable à la tabulation. Toutefois, dans certaines situations, ces éléments peuvent être ajoutés au flux de la tabulation au clavier :

- Possibilité d'atteindre un code uniquement atteignable à la souris car généré en JavaScript plutôt qu'en html natif.
- Possibilité d'atteindre des composants ARIA/Javascripts comme par exemple le tab panel (<li role= « tab » tabindex= « 0 »).
- Forcer le lecteur d'écran à lire du texte dans un formulaire qui risquerait d'être skippé.

Tabindex= « -1 » pour rendre un élément focusable mais pas tabulable

Ce paramètre ne permet pas de tabuler jusqu'à l'élément mais de lui envoyer le focus. Cette technique peut être utile dans plusieurs situations :

- Envoyer le focus vers un message (par exemple un message d'erreur) pour s'assurer que les personnes voyantes et utilisatrices d'un lecteur d'écran le voient et l'entendent.
- Envoyer le focus dans un composant en JavaScript.
- Dans un lien d'évitement. Certains navigateurs, comme Safari, nécessitent un élément de destination focusable nativement (lien, bouton…) ou qui contient une valeur de tabindex. Sinon, le viewport scrollera vers la position voulue mais la tabulation suivante positionnera le focus sur l'élément qui suit directement le lien d'évitement. L'ajout d'un tabindex= « -1 » sur l'élément de destination y positionne le focus sans interférer avec l'ordre éventuel de tabulation présent sur la page.

Ne pas utiliser de tabindex positif

Un élément avec un tabindex positif recevra le focus en premier. Le flux normal sur la page en sera complètement perturbé et les contenus dynamiques pourraient ne plus fonctionner correctement.

Q. Images

Tous les contenus non textuels nécessitent une alternative en texte. Les utilisateurs de lecteurs d'écran ne peuvent pas lire une image, les personnes déficientes visuelles ont besoin de (dé)zoomer, modifier la couleur, le sens de certaines images n'est pas toujours compris…

Aussi, tant les images, les boutons, les vidéos, les fichiers audio… doivent présenter une alternative ou une traduction en texte que tout le monde pourra lire et adapter à ses besoins.

Les alternatives d'images

Les images non décoratives doivent fournir une alternative techniquement associée, explicite et concise (150 caractères est le tout grand maximum).

Si l'image nécessite beaucoup d'explications (comme une infographie par exemple), l'alternative ne doit pas nécessairement y être associée mais doit être présente sur la page. L'image devient alors de la décoration sans alternative textuelle.

L'objectif est de permettre de donner la même information aux personnes aveugles qu'aux voyants, sans pour autant polluer leur lecture avec trop d'informations inutiles ou redondantes.

a) Les images informatives

Les images informatives apportent de l'information non visible dans la page. Si on enlève cette image, on perd la compréhension de tout ou partie de la page.

Les images qui apportent du contenu doivent contenir une alternative explicite techniquement associée

Les lecteurs d'écran ne peuvent pas (encore) déduire le sens de l'image. Ils lisent le contenu de l'attribut alt de la balise <img>.

Bon exemple : Une image qui contient une alternative dans l'attribut alt

Partant du postulat que, dans son contexte, l'image suivante apporte une information nécessaire à la compréhension de la page, son alternative pourrait être « A Singer model antique sewing machine ».

En passant sur l'image, le lecteur d'écran dira « Graphique (ou image), A Singer model antique sewing machine ».

L'utilisateur saura qu'il s'agit d'une image (pas besoin de lui rappeler dans le texte) et entendra l'alternative textuelle.

<img src="sewing_machine_singer.jpg" width="686" height="518"

alt="A Singer model antique sewing machine">

Mauvais exemple : Une image qui ne contient pas d'attribut alt

Attention, si l'image ne contient pas d'attribut alt, le lecteur d'écran tentera de déduire l'alternative et lira le nom de l'image et / ou son chemin. Ce qui n'est pas confortable à l'écoute et peut générer des erreurs de compréhension. Dans le cas ci-dessous, le lecteur d'écran dira « Graphique, sewmach2 dot J P G ».

<img src="sewmach2.jpg" width="686" height="518">

L'alternative de l'image doit être explicite et liée à son contexte et à son utilité dans la page où elle se trouve.

Les éléments suivants doivent être pris en compte :

- L'alternative de la même image peut varier en fonction de son contexte. Sur l'exemple d'un logo : donner le nom de la marque ou, s'il s'agit d'une image lien vers l'accueil du site, renseigner « accueil du site ».
- L'alternative ne peut pas être générique. Sur l'exemple d'une image d'un tampon qui valide une information, l'alternative n'est pas « tampon à encre » mais « Validé ».
- Ne pas reprendre les mots « image de… » ou « photo de… » puisque le rôle d'image est clairement reconnu par le lecteur d'écran qui dira cette information avant d'en lire l'alternative.
- Maximum 150 caractères.
- Utiliser la ponctuation.
- Ne pas renseigner de mots clés pour le référencement (SEO).
- Ne pas renseigner dans l'attribut alt une information qui doit être vue par tous les internautes. Exemple : un copyright, l'auteur d'une photo, une légende…

b) Les images de décoration ou redondantes.

Une image de décoration ou redondante n'apporte aucune information complémentaire au contenu de la page. Sont concernées :

- Des images ajoutées pour le design de la page, pour donner une « ambiance », pour attirer le regard…
- Des images qui apportent du contenu qui est déjà présent sur la page (Exemple : une image de graphe qui illustre un tableau de données correctement structuré en html).

Pour décider si une image rentre dans cette catégorie, il faut imaginer la page sans l'image et voir si on comprend toujours toute la page sans perdre d'information. Il s'avère que les images sont souvent des images de décoration et qu'elles ne demandent donc pas d'alternative textuelle.

Les images de décoration ou qui apporte du contenu redondant sur la page doivent avoir soit une alternative vide alt= « », soit un attribut aria-hidden= « true », soit un role ARIA role= « presentation » ou être implémentées comme background en CSS

Le but est de permettre à l'utilisateur d'un lecteur d'écran de lire le plus rapidement possible sans être pollué par de l'information inutile.

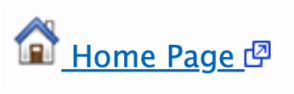
Avec ces techniques, la plupart des lecteurs d'écran ne disent même pas à son utilisateur que l'image est présente sur la page (ce qui est bien le but recherché).

Attention à bien ajouter l'attribut alt vide alt= « ». Sans cet attribut, le lecteur d'écran tente de déduire l'alternative en lisant le nom et / ou le chemin de l'image.

Bon exemple : Une image redondante avec un alt vide

Dans cet exemple, le lien vers la page d'accueil contient une image et un texte explicite « Home Page ». L'alternative de l'image doit donc être vide.

Par contre, si le lien ne contenait pas le texte « Home Page », l'alternative de l'image devrait être alt= « Home Page ».




```
<a href="https://eqla.be">
 <img src="home-icon.png" width="24" height="25" alt="">
 Home Page
</a>
```

c) Les images actionnables (liens, boutons, contrôles…)

Les images qui servent de liens, boutons ou autres contrôles doivent avoir une alternative qui répond aux mêmes critères que ceux décrits pour les types d'images précédents.

On évitera donc les alternatives du type « lien vers… », « logo Eqla » plutôt que « Retour à l'accueil », alternatives vides…

Les boutons de formulaires de type input type= « image » doivent respecter les mêmes consignes et afficher une alternative qui reprend le nom sur l'image.


```
<input type="image" name="submit" src="submit-button.png" alt="Submit">
```

L'alternative doit être explicite quant à sa fonction et sa destination.

d) Les images complexes

Les images complexes sont celles qui ne peuvent être décrites de manière explicite et complète en moins de 150 caractères.

Les images complexes doivent être brièvement décrites via l'attribut alt ET doivent proposer une description complète.

Il existe plusieurs techniques pour fournir une description longue à une image complexe. En fonction du site, de la charte graphique, de la place sur la page… une des techniques ci-dessous pourra être utilisée :

- Donner la description complète dans le texte de la page lui-même (En fonction des cas, l'attribut alt pourra rester vide).
- Prévoir un bouton (accordéon) sous l'image qui affiche et cache une région qui contient la description longue.
- Prévoir un bouton qui ouvre une boite de dialogue qui contient la description longue
- Prévoir un lien vers la description longue écrite en html sur une autre page
- Intégrer l'image dans une balise <figure> où la description est reprise dans la balise <figcaption> intégrée entre les balises d'ouverture et de fermeture <figure>.

Les liens, boutons… doivent être visibles pour tous, explicites et techniquement associés.

Bon exemple : Une longue description avec aria-describedby dans le contexte du document

Dans cet exemple : l'image apparaît et son alternative alt est lue (pourrait être vide si le texte est repris par ailleurs) puis la description complète « Last year… » associée à l'image est lue également.

<h1> Proportion of Josephine's Mealtime Squirrel Sightings, by Month</h1>

<img class="border" src="bar-chart.png" width="546" height="330" alt="Bar chart with percentages. Extended description below chart." aria-describedby="description-extended">

Last year, Josephine kept track of the number of times…

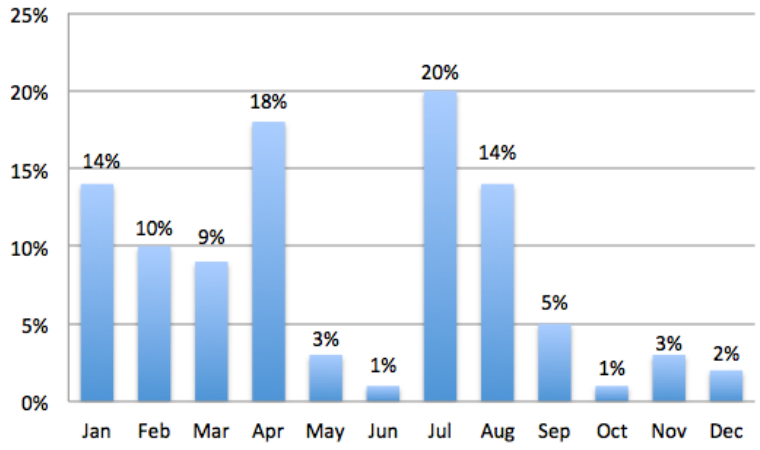
January: 14%

February: 10%

…

…

Proportion of Josephine's Mealtime Squirrel Sightings, by Month



Last year, Josephine kept track of the number of times…

- January: 14%
- February: 10%

e) Les images-texte

Une image-texte représente du texte sous forme d'image. Soit l'image représente uniquement le texte (un logo par exemple), soir du texte en image est ajouté à une image ou sur un fond.

Ce type d'image ne devrait pouvoir être utilisé que dans le cas de logos ou pour un usage similaire.

Elles ne sont pas lisibles par les lecteurs d'écran et une alternative textuelle doit donc être prévue. Par ailleurs, les personnes qui doivent (dé)-zoomer leur écran risquent de perdre la

compréhension du texte de par la pixélisation (sauf pour les images en SVG). Enfin, les daltoniens qui doivent changer la couleur du texte ou les personnes dyslexiques qui préfèrent appliquer leur police spécifique ne pourront pas le faire.

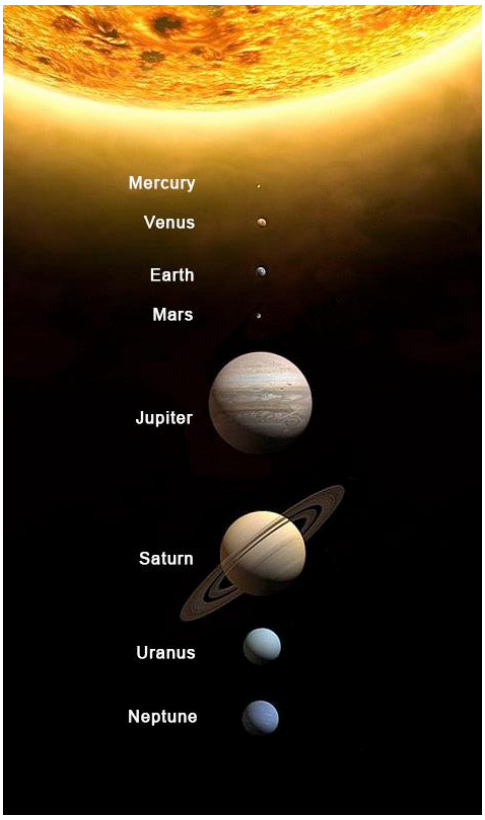
f) Les images Maps

Les images Maps peuvent être rendues accessibles mais elles doivent respecter une série de critères dont beaucoup ont déjà été décrits dans les autres types d'images.

- Les alternatives doivent être techniquement associées et décrire suffisamment l'image de manière explicite
- Le nombre de caractères des alternatives ne peut être supérieur à 150
- Les alternatives des balises <area> doivent respecter toutes les mêmes consignes que celles vues précédemment
- …

Bon exemple d'image Maps : le système solaire dont les planètes pointent vers un autre site

```
<img src="solar_system.jpg" alt="Solar System" width="472" height="800" usemap="#Map3">
 <map name="Map3">
  <area shape="rect" coords="115,158,276,192"
  href="http://en.wikipedia.org/wiki/Mercury_%28planet%29" target="_blank" alt="Mercury (Wikipedia article)">
  <area shape="rect" coords="115,193,276,234"
  href="http://en.wikipedia.org/wiki/Venus" target="_blank" alt="Venus (Wikipedia article)">
  <area shape="rect" coords="118,235,273,280"
  href="http://en.wikipedia.org/wiki/Earth" target="_blank" alt="Earth (Wikipedia article)">
  <area shape="rect" coords="119,280,272,323"
  href="http://en.wikipedia.org/wiki/Mars" target="_blank" alt="Mars (Wikipedia article)">
  <area shape="rect" coords="119,324,322,455"
  href="http://en.wikipedia.org/wiki/Jupiter" target="_blank" alt="Jupiter (Wikipedia article)">
  <area shape="rect" coords="118,457,352,605"
  href="http://en.wikipedia.org/wiki/Saturn" target="_blank" alt="Saturn (Wikipedia article)">
  <area shape="rect" coords="119,606,308,666"
  href="http://en.wikipedia.org/wiki/Uranus" target="_blank" alt="Uranus (Wikipedia article)">
  <area shape="rect" coords="117,664,305,732"
  href="http://en.wikipedia.org/wiki/Neptune" target="_blank" alt="Neptune (Wikipedia article)">
 </map>
```



Les images SVG

Toutes les informations et en particulier celles relatives à l'accessibilité des SVG est disponible en ligne, notamment sur css-tricks.com

Le format SVG est intéressant en accessibilité de par son poids léger et sa capacité à s'agrandir sans perte de qualité. Les personnes qui doivent zoomer leur écran pourront toujours garder la même qualité d'image.

Comme pour tous les types d'images, certains critères d'accessibilité doivent être pris en compte pour s'assurer de communiquer l'information à toutes les personnes.

Il existe plusieurs moyens d'intégrer du SVG dans une page web :

- Utiliser la balise <img> en y référençant sa source. Exemple : <img src= « svgfile.svg » alt= « alternative text »>.
- Intégrer le SVG directement (inline) dans le code source HTML avec l'élément <svg>
- Intégrer le SVG dans une balise <iframe> ou <embed> ou le référencer comme attribut dans un <object>.

En accessibilité, nous privilégions les 2 premières méthodes car les lecteurs d'écran présentent des failles dans la lecture des SVG dans <iframe> et <object>.

a) Les SVG intégrés avec la balise <img> devraient avoir un role= « img » et doivent (sauf déco) avoir une alternative explicite et concise via alt, aria-label ou aria-labelledby

Le role= « img » garantit que le lecteur d'écran reconnaîtra l'image comme telle et nous ne revenons plus sur l'importance des alternatives.

Bons exemples de SVG implémentés avec la balise <img> et une bonne alternative
<img src="somesvg.svg" role="img" alt="a concise description of the image">

<img src="somesvg.svg" role="img" alt="a concise description of the image">

This is a caption above an image

<img src="somesvg.svg" role="img" aria-labelledby="caption1">

b) Éléments obligatoires en accessibilité pour les SVG inline intégrés avec la balise <svg>

- La balise <svg> doit contenir le role= « img »
- Les SVG informatifs ou actionnables doivent avoir une alternative textuelle dans un élément <title> qui doit être le premier enfant du SVG
- L'alternative dans le <title> doit être techniquement associé à l'élément <svg> avec aria-labelledby (pour une compatibilité maximum avec tous les lecteurs d'écrans)
- Tous les textes dans l'image qui doivent être lus par le lecteur d'écran doivent être associés au <svg> avec l'élément aria-labelledby

Exemple d'un extrait d'une image svg inline tous ces éléments

<svg role="img" aria-labelledby="title desc jan feb mar apr may jun jul aug sep oct nov dec">

<title id="title">Total Widgets Purchased during 2016</title>

<desc id="desc">

The graph displays the total number of widgets purchased from The ABC Store

during 2016, displayed by month.

</desc>

...

<g id="jan" class="bar labels x-labels">

<rect x="25" y="195" width="33" height="45" fill="#111"></rect>

<text x="32" y="260" fill="#000">Jan.</text>

<text x="33" y="220" fill="#fff">45</text>

</g>

<g id="feb" class="bar labels x-labels">

<rect x="62" y="160" width="33" height="80" fill="#111"></rect>

<text x="70" y="260" fill="#000" >Feb.</text>

<text x="71" y="220" fill="#fff">80</text>

</g>

<g id="mar" class="bar labels x-labels">

<rect x="99" y="140" width="33" height="100" fill="#111"></rect>

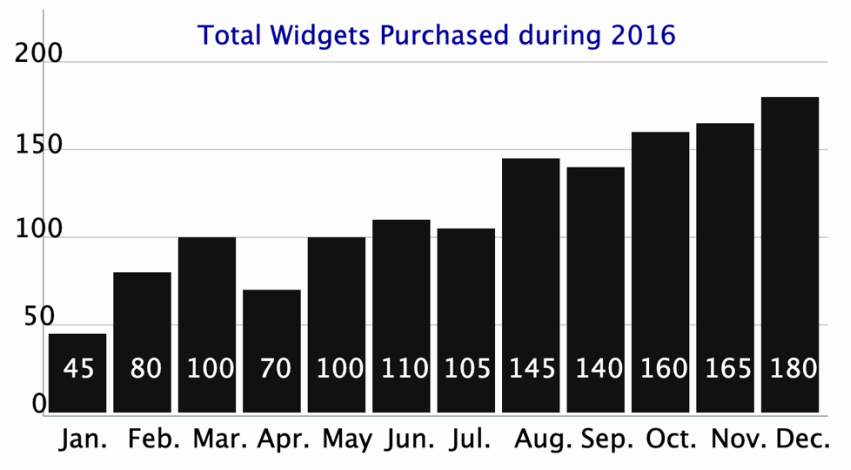
<text x="107" y="260" fill="#000" >Mar.</text>

<text x="103" y="220" fill="#fff">100</text>

</g>

...

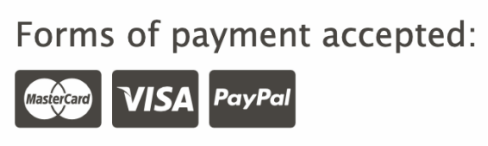
</svg>



Les Icon Fonts

a) *Les Icon Fonts informatives*

Les Icon Fonts sans texte visible qui apportent de l'information doivent avoir un role= « img » et un texte alternatif



Forms of payment accepted:

Les Icon Fonts actionnables sans texte visible doivent avoir un texte alternatif

Bon exemple : Des boutons icon fonts avec une alternative via aria-label et un role= « img »

<button></button>

<button></button>

<button></button>

<button></button>

<button></button>

<button></button>

```
  <button><span role="img" class="fa fa-unlink" aria-label="Remove link"></span></button>
  <button><span role="img" class="fa fa-list" aria-label="Bulleted list"></span></button>
  <button><span role="img" class="fa fa-list-ol" aria-label="Numbered list"></span></button>
</p>
```

Bon exemple : Un lien icon fonts avec une alternative via aria-label

Dans cet exemple, le texte alternatif est placé via l'attribut aria-label. Les lecteurs d'écran traitent l'icône comme une image et le lien hérite de l'alternative de l'icon font.

```
<p>
  <a href="#">
    <span class="fa fa-facebook-official fa-2x" aria-label="Our Facebook page">
    </span>
  </a>
</p>
```

b) *Les Icon Fonts décoratives ou redondantes*

Cette notion a été expliquée dans le chapitre relatif aux alternatives d'images

Les icon fonts concernées doivent être cachées aux lecteurs d'écran via l'attribut aria-hidden= « true »

Bon exemple : Cacher une icône d'aide à côté d'un texte d'aide


```
<p>
  <a href="#">
    Help
    <span class="fa fa-question-circle" aria-hidden="true"></span>
  </a>
</p>
```

IV. Conclusion

Alors que la crise du Covid a, sans aucun doute, accéléré la numérisation des services publics, 40% de la population belge sont toujours en situation de vulnérabilité face à la numérisation croissante de la société[6].

Pour répondre à ce défi, et afin que les services publics numérisés soient réellement inclusifs et accessibles, cette étude a permis d'établir qu'ils ne pouvaient justement pas être exclusivement numériques.

Malgré tous les efforts qui pourraient être mis en œuvre, une partie importante de la population restera encore pour longtemps en besoin de contact humain pour réaliser ses démarches administratives. En effet, « *près de six internautes sur dix âgés de 16 à 74 ans (58 %) n'ont au mieux qu'un recours limité à internet pour effectuer leurs démarches administratives : 32 % d'entre eux n'effectuent aucune démarche par voie numérique, 26 % n'y recourent que de façon limitée* »[7].

Un autre constat est que la numérisation des démarches reporte la charge de l'accompagnement du citoyen vers les assistants sociaux, les services d'accompagnement, voire même des informaticiens (dont le métier n'est pas de former à l'utilisation du lecteur de carte d'identité ou encore d'effectuer des virements bancaires en ligne pour le compte de leurs clients).

Par ailleurs, l'étude a également démontré que les normes d'accessibilité (obligatoires pour les services publics) sont un socle solide, mais insuffisant pour la prise en compte de l'ensemble des utilisateurs des services numérisés. Il est dès lors indispensable de mettre en œuvre les 38 recommandations reprises dans ce rapport et de faire tester par un panel composé de différents publics, dès l'étape de conception, les plateformes des services numérisés. Cette recommandation rejoint les travaux antérieurs des décrits dans le rapport d'Idéalic[8] et dans la brochure sur l'inclusion numérique[9] du catalogue BOSA.

Afin d'implémenter ces recommandations, un référentiel technique est mis à disposition des concepteurs pour en assurer la mise en œuvre.

Enfin, il y a lieu de mettre en place des procédures d'accompagnement, de formation et de contrôle de la conformité des projets à ces recommandations pour garantir la prise en compte de l'ensemble des publics dans le développement des outils numériques régionaux.

---

[6] Fondation Roi Bauduin, (2020). Baromètre de l'inclusion numérique 2020

[7] Fondation Roi Bauduin, (2021). *Inclusion numérique : les services numériques essentiels : profitables à toutes les personnes ?* p24. En ligne sur : https://www.kbs-frb.be/fr/inclusion-numerique-les-services-numeriques-essentiels-profitables-toutes-les-personnes

[8] Faure, L & Brotcorne, P (2021). *Guide pour une conception inclusive des services numériques.* Idéalic.be. p.20-24.

[9] L'inclusion numérique, qu'est-ce que c'est ? En ligne sur : https://digitalopen.belgium.be/fr/playbook/tools/linclusion-numerique-brochure

V. Check-list pour des SPN accessibles et inclusifs à destination des services publics

- ✓ Recommandation 1 : En complément aux services en ligne, et pour toute demande, les citoyens ont la possibilité de s'adresser à un guichet physique, en présence d'un agent qui pourra les aider à réaliser leurs démarches
- ✓ Recommandation 2 : Le site respecte les normes d'accessibilité numérique et contient une déclaration d'accessibilité conforme, rédigée ou validée par un organisme compétent
- ✓ Recommandation 3 : Uniformiser la mise en page de tous les services publics numérisés
- ✓ Recommandation 4 : La couleur d'un texte doit être suffisamment contrastée avec la couleur de fond
- ✓ Recommandation 5 : Utiliser une police accessible et veiller à ce que les tailles de caractères ne soient pas figées
- ✓ Recommandation 6 : L'information ne doit pas être donnée uniquement par la couleur ou la forme
- ✓ Recommandation 7 : Adopter le Responsive Design
- ✓ Recommandation 8 : Prévoir au moins les 3 chemins d'accès suivants pour naviguer : le menu de navigation, le plan de site et un moteur de recherche
- ✓ Recommandation 9 : Prévoir un moteur de recherche performant et utilisable à la voix
- ✓ Recommandation 10 : Respecter les standards de structuration des sites internet
- ✓ Recommandation 11 : Implémenter les liens d'évitement
- ✓ Recommandation 12 : Afficher une information permettant de connaître son emplacement dans l'arborescence du site
- ✓ Recommandation 13 : L'entièreté du site doit être utilisable au clavier
- ✓ Recommandation 14 : Utiliser des termes explicites qui décrivent précisément l'action d'un élément (lien/bouton)
- ✓ Recommandation 15 : Limiter la longueur des textes et illustrer le contenu avec des visuels
- ✓ Recommandation 16 : Traduire le site en français, néerlandais et anglais
- ✓ Recommandation 17 : La langue doit rester cohérente pour l'utilisateur tout au long de la navigation
- ✓ Recommandation 18 : Traduire les actions administratives en FALC et en Langue des Signes
- ✓ Recommandation 19 : Ajouter une alternative textuelle pour les images qui contiennent une information
- ✓ Recommandation 20 : Mettre en œuvre la procédure de connexion via CSAM
- ✓ Recommandation 21 : Prévoir un message qui confirme l'état de la connexion/déconnexion
- ✓ Recommandation 22 : Prévoir un message d'alerte si un utilisateur ferme une fenêtre sans s'être déconnecté au préalable
- ✓ Recommandation 23 : Allonger la durée maximale pour la procédure de connexion
- ✓ Recommandation 24 : Assurer l'accessibilité des formulaires
- ✓ Recommandation 25 : Demander uniquement les informations strictement nécessaires
- ✓ Recommandation 26 : Permettre d'introduire et de modifier manuellement ses données personnelles
- ✓ Recommandation 27 : Préciser le format des informations demandées
- ✓ Recommandation 28 : En cas d'erreur, indiquer clairement l'information à modifier

- ✓ Recommandation 29 : Prévoir un message qui confirme l'état de soumission du formulaire
- ✓ Recommandation 30 : Identifier la nature et le poids du document téléchargeable
- ✓ Recommandation 31 : Expliquer la numérisation et le téléversement des documents
- ✓ Recommandation 32 : Proposer le paiement par carte de débit (Bancontact) et de crédit (Visa et Mastercard)
- ✓ Recommandation 33 : Prévoir un message qui confirme l'état du paiement en ligne
- ✓ Recommandation 34 : Les vidéos et fichiers sonores doivent systématiquement être accessibles à tous
- ✓ Recommandation 35 : La prise de rendez-vous doit pouvoir s'effectuer par téléphone et via un calendrier en ligne
- ✓ Recommandation 36 : Proposer au minimum le mail ou formulaire de contact accessible et le téléphone (complété d'un dispositif d'interprétation à distance en langue des signes)
- ✓ Recommandation 37 : Faire tester le SPN par un panel diversifié de citoyens dès la conception
- ✓ Recommandation 38 : Réaliser un mode d'emploi des SPN
- ✓ Recommandation 39 : Former les professionnels aux recommandations inclusives
- ✓ Recommandation 40 : Mettre à disposition des organes d'accompagnement et de formation une plateforme test en ligne permettant aux utilisateurs d'essayer les services en ligne lors de formations

Recommandations inclusives et référentiel technique permettant d'améliorer et d'optimiser les services publics numérisés pour les personnes présentant un risque de fracture numérique

CAWab pour Paradigm