L’accessibilité commence dans la planification : comment développer des sites Web accessibles?

Conférence en ligne de Cédric Anderson (Laboratoire NT2)
10 juin 2020

Cette conférence s’inscrit dans le cadre d’*Interroger l’accès*, une série de conférences et d’ateliers sur l’accessibilité des productions artistique et médiatique, développée par OBORO et Spectrum Productions avec le soutien du Conseil des arts du Canada. OBORO et Spectrum Productions reconnaissent que leurs activités ont lieu à Tiohtià:ke, en territoire kanien’kehá:ka non cédé.

(Début de la transcription)

Bonjour. Mon nom est Cédric Anderson et nous allons voir ensemble aujourd’hui ce qu’est l’accessibilité Web au Québec. Avant de commencer, j’aimerais remercier le centre OBORO de m’avoir invité et j’espère que la présentation va vous plaire.

À l’ordre du jour :

- Qui je suis et ce que je fais comme travail.
- L’état des lieux de l’accessibilité Web au Québec.
- Le WCAG 2.0, qui est le protocole utilisé au Canada.
- WAI-ARIA, qui sont des aides pour les balises HTML.
- Le HTML accessible.
- Les outils en ligne qui nous permettent de vérifier qu’un site est bel et

bien accessible.
- L’accessibilité Web selon certains cas de figure.
- Je travaille dans le domaine du Web depuis 2003. Je travaille comme développeur « Full Stack », donc je m’occupe de l’avant-scène et de l’arrière-scène lorsqu’on fait un site Internet. Je travaille beaucoup avec des institutions publiques et privées, des petites entreprises, des artistes, des travailleurs autonomes. Je travaille beaucoup avec WordPress. Je fais des thèmes personnalisés pour la plateforme WordPress. Et je travaille actuellement à l’UQAM, au laboratoire NT2, qui est un laboratoire de recherche en arts et en littérature numériques.

Je me dois de vous lire la définition de l’accessibilité Web :

« Un contenu ou un service offert sur le Web est considéré comme accessible lorsque toute personne, peu importe ses incapacités, peut le comprendre, y naviguer et interagir avec lui. L’accessibilité Web représente un élément important pour permettre l’intégration sociale et la participation à la vie collective de la population. »

Il y a une référence dans le bas de la diapositive si vous souhaitez aller voir le site Web du gouvernement.

Au Québec, présentement, il n’y a pas grand monde qui doit suivre les règles d’accessibilité. Mis à part le gouvernement, il n’y a presque personne. Il y a les collèges et les établissements universitaires, mais dans la majorité des cas, ce sont les ministères et les organismes gouvernementaux qui se doivent d’être accessibles. Ces deux diapositives avec des liens dans le bas vont vous expliquer exactement qui doit appliquer les règles d’accessibilité. Mais sachez que si vous êtes des artistes, un centre d’artistes, une petite entreprise, un OSBL, vous n’êtes pas obligés d’avoir un site accessible.

Le WCAG 2.0, c’est un ensemble de règles qui nous permet de s’assurer que le site va être accessible par la majorité des gens. Il y a beaucoup, beaucoup de règles. C’est un très gros document. Par contre, on peut l’expliquer en quatre points : un site se doit d’être perceptible, utilisable, compréhensible et robuste. Vous allez trouver le lien à la fin du document qui explique exactement ce qu’est le WCAG.

Le WAI-ARIA, c'est des propriétés qu'on ajoute à nos balises HTML pour mettre les contenus en contexte. Pour les gens qui développent des sites Web, lorsqu'on crée une boîte de contenu, on peut seulement dire « ceci est une boîte », mais il est difficile d'expliquer exactement qu'est-ce que le contenu de cette boîte. Le WAI-ARIA nous amène une série de propriétés HTML qui nous permettent de spécifier les contenus de chacune des boîtes dans nos pages Web, donc de dire : « ceci, ce sont des heures d'ouverture » ou « ça, c'est une navigation », par exemple. Donc le WAI-ARIA nous permet de déterminer des rôles, des propriétés, des états. Les propriétés ARIA devraient être utilisées seulement lorsque c'est nécessaire. Il y a des balises HTML qu'on appelle les balises sémantiques. Elles peuvent déjà nous donner une idée large des contenus. Il y a les balises, par exemple : NAV, ARTICLE, FOOTER (pour le pied de page). Donc les balises sémantiques doivent passer avant les règles WAI-ARIA.

Nous allons voir ensemble les techniques qui sont à notre disposition pour créer un site Web accessible.

La première chose, comme je le disais, ce sont les balises sémantiques. Elles permettent de mettre le contenu en contexte. Elles aident à la navigation au clavier. Pour les gens par exemple qui ont des troubles moteurs ou qui utilisent un lecteur d'écran, c'est beaucoup plus facile de naviguer sans souris, seulement avec le clavier.

C'est beaucoup plus simple pour les développeurs. Lorsqu'on développe un site Internet, utiliser des balises sémantiques est très simple. C'est plus simple pour les sites « responsives » – les sites qui fonctionnent autant sur un écran d'ordinateur que sur un écran mobile. Et ça aide aussi à l'optimisation pour les moteurs de recherche. Google aime beaucoup qu'on fasse des sites Web sémantiques et va s'assurer qu'on a de meilleurs résultats si on utilise les bonnes balises.

Pour ce qui est des textes dans les pages, il n'y a pas nécessairement de règles, mais c'est d'y aller logiquement. On doit avoir un langage clair, par exemple, utiliser « 5 à 7 » et non pas le slash, « 5/7 », qui ne veut rien dire pour un lecteur d'écran. On ne va pas utiliser d'abréviation, mais si on doit utiliser des abréviations, on doit utiliser la balise ABBR, qui est la balise « abréviation ».

Utiliser une grosseur de police adéquate, le minimum étant 16 pixels. En bas de 16 pixels, ça commence à faire des textes très petits. Pour les gens qui ont des troubles de vision, ça peut être difficile. On doit s'assurer que la couleur des textes soit bien visible par rapport à la couleur de fond. Par exemple, sur cette présentation, c'est un texte blanc sur un fond gris très foncé, donc c'est facile à lire. Par contre, le bleu de « TEXTE » est plus difficile à lire pour certaines personnes.

Utiliser les bonnes balises. On voit souvent des cas d'erreur où les gens vont utiliser des balises qu'ils ne devraient pas utiliser. On devrait utiliser les bonnes balises pour les bonnes choses. Par exemple, pour l'en-tête d'une page, utiliser les balises H1, H2, H3, à la place d'utiliser une balise « paragraphe ». Et ne pas utiliser les attributs HTML pour le design. C'est vraiment de séparer le visuel et le contenu – le contenu est géré par le HTML, le visuel est géré par le CSS.

Pour les hyperliens (les liens dans nos pages) : on doit s'assurer que le texte du lien soit court, simple, clair et descriptif. Par exemple, un lien qui dit « Cliquez ici », ça ne veut rien dire hors contexte. On devrait utiliser à la place « Pour l'achat d'un billet, veuillez cliquer ici », par exemple. C'est tout à fait correct d'utiliser des phrases complètes en tant que liens.

Lorsque c'est possible, ne pas utiliser le même texte de lien plusieurs fois dans la même page. Pour ceux qui ne connaissent pas les lecteurs d'écran, ils sont utilisés principalement par les gens qui ont des troubles de vision ou qui sont aveugles. Le lecteur d'écran est une voix qui décrit ce qu'il y a dans la page, ligne par ligne. Vous pouvez imaginer que si j'ai cinq boutons « Cliquez ici », ça va être difficile de savoir exactement lequel fait quoi. Normalement, on devrait utiliser des textes différents pour chacun des liens dans nos pages.

Utiliser l'attribut TITLE lorsque c'est possible. L'attribut Title, c'est pour les images, les hyperliens. Ce n'est pas visible par les lecteurs d'écran, donc il ne faut pas mettre de texte important. L'attribut Title, c'est lorsqu'on passe avec la souris sur un lien et il y va y avoir une petite bulle explicative – c'est l'attribut Title. Donc, s'assurer de mettre une description qui explique bien où le lien s'en va.

Lorsqu'on écrit un lien vers un fichier, écrire le poids de celui-ci, et utiliser

l’attribut DOWNLOAD. Il y a un attribut HTML qui est « download » et on devrait l’utiliser. On devrait aussi, si possible, écrire le poids du fichier qu’on est supposé télécharger. Ça va aider les gens à savoir s’ils ont les ressources nécessaires, par exemple.

Ne pas répéter l’URL dans le texte du lien. Par exemple, si je fais un lien vers google.com, je ne vais pas écrire « www.google.com » comme texte. À la place, je vais écrire par exemple : « Cliquez ici pour le moteur de recherche Google ». Ce serait un meilleur texte.

Mettre une mention lorsque le lien s’ouvre dans un autre onglet. Les lecteurs d’écran ne vont pas avertir la personne qui utilise le lecteur d’écran qu’un nouvel onglet s’est ouvert. Donc c’est bien de mentionner dans le lien que le lien va s’ouvrir dans un nouvel onglet.

Et utiliser les « skip links ». C’est très important, on va le voir un peu plus tard.

TABINDEX : lorsqu’une personne utilise le clavier pour naviguer dans un site, il va utiliser la touche Tab beaucoup. C’est la touche qui nous permet de passer d’un élément à l’autre dans une page Internet, beaucoup plus que les flèches, par exemple.

Le Tabindex nous permet de changer le flux normal de navigation dans une page. Imaginez une navigation dans le haut de la page où il y a un menu et quand on passe sur « À propos », il y a un sous-menu qui s’ouvre. Si le flux normal de navigation lorsque j’appuie sur Tab est : Accueil > À propos > Produits > Services > Joindre, par exemple, mais lorsque la personne arrive sur À propos, au lieu de passer au prochain, j’aimerais qu’elle aille dans le sous-menu, je peux changer le Tabindex. Donc si je fais la modification dans le Tabindex, la personne va appuyer sur Tab, ça va faire Accueil > À propos > de l’équipe > de l’entreprise > Services > Joindre, etc.
Le Tabindex permet de rendre un élément accessible via la navigation au clavier. Normalement, lorsque j’appuie sur Tab, ça va sélectionner les liens, principalement. Si j’appuie sur Tab, ça va passer sur la navigation. Une fois au bout, ça va m’amener au prochain lien dans la page, etc. Des fois, on voudrait que les personnes puissent accéder à un élément dans la page et c’est possible en mettant la propriété Tabindex à zéro. Donc la section que j’ai mise comme Tabindex="0" va être visible par le Tabindex. Et ça me

permet aussi de faire l'inverse : de rendre un élément non accessible via la navigation au clavier, en mettant la propriété Tabindex à -1.

Les images : on doit toujours utiliser l'attribut ALT. Si j'utilise la balise IMG, ça devrait devenir un automatisme d'écrire IMG (espace) ALT. L'attribut ALT, c'est le texte alternatif pour une image. Donc c'est une petite description de ce que représente l'image. Par exemple : « Panneau de signalisation disant de tourner à gauche ». Ou ça pourrait être : « Œuvre d'art/peinture/huile », etc. Donc une phrase ou deux mini-phrases qui expliquent bien ce qu'il y a dans l'image.

On doit utiliser l'attribut ALT vide pour les images décoratives. Donc toutes les images qui ne sont pas nécessaires au contenu, qui sont là seulement purement esthétiquement. On devrait tout de même mettre la propriété ALT, mais on devrait la mettre vide. Comme ça, les lecteurs d'écran vont passer à côté et ne vont pas le lire.

L'attribut Title pour les bulles d'aide : j'ai parlé tout à l'heure de l'attribut Title.

Utiliser l'attribut LONGDESC (qui veut dire « longue description ») lorsqu'on utilise des schémas ou des images complexes. Il faut se rappeler par contre que ce n'est pas visible par tous les lecteurs d'écran, mais la majorité sont capables de les lire. Les longues descriptions, on va les utiliser lorsque j'ai un schéma qui représente une statistique, par exemple, ou les schémas en pointes de tarte. C'est difficile à expliquer en une phrase. De mettre une seule phrase, ça va être l'attribut ALT, qui dit « Schéma de statistiques des visites sur le site de 2019 », par exemple. Ça ne donne pas la statistique à la personne qui est malvoyante. Donc en utilisant la longue description, ça nous permet de mettre un lien vers une description longue. La personne, quand elle va avoir accès à cette longue description, devrait avoir accès à une description complète qui explique que 10 % est attribué à telle chose, 25 % à telle chose, dans le cas, par exemple, de statistiques.

Utiliser des noms de fichier descriptifs : si j'ai une image .JPEG, par exemple, il y a certains lecteurs d'écrans qui vont lire les noms des fichiers. Donc c'est une bonne idée de mettre des noms des fichiers qui soient descriptifs, à la place d'avoir seulement « img001.jpeg ».

Si la description d'une image est dans un paragraphe adjacent, utiliser l'attribut ARIA-LABELEDBY. Tout à l'heure, on parlait de ARIA. Ceci est un attribut qui nous permet, par exemple, de dire que cette image-là, le paragraphe est juste à côté et c'est la description de cette image.

Et on devrait utiliser les balises FIGURE et FIG CAPTION qui sont des balises sémantiques qui nous permettent de dire : voici une figure qui contient une image, une description, un paragraphe et un MP3, par exemple. Ça nous permet de regrouper plusieurs éléments qui font partie de cette image.

Pour les formulaires : on doit s'assurer que nos formulaires soient simples, clairs et logiques. On voit souvent des sites, par exemple, où il y a des formulaires sur plusieurs pages : section 1, section 2, section 3 avec des « Suivant ». C'est déjà frustrant pour quelqu'un qui n'a pas un handicap visuel ou de problème de vision. Je n'ose pas imaginer pour quelqu'un qui utilise un lecteur d'écran.

S'assurer qu'ils fonctionnent avec une navigation au clavier en utilisant le Tabindex. C'est un excellent exemple, les formulaires, pour le Tabindex que j'expliquais juste avant. Par exemple, si je fais un formulaire sur deux colonnes, normalement le Tab va y aller la colonne de gauche en premier, ensuite la colonne de droite. Mais peut-être que je voudrais, par exemple, qu'il fasse de gauche à droite, gauche à droite, gauche à droite. Je vais pouvoir contrôler ça avec le Tabindex.

Utiliser les LABEL, ou bien ARIA-LABEL : une autre erreur qu'on voit souvent sur les sites, c'est que les gens... esthétiquement, c'est plate d'avoir, par exemple, un champ « Nom » et que ce soit écrit « Nom » au-dessus. Les gens préfèrent mettre le nom à l'intérieur du champ. Par défaut, on doit utiliser un « placeholder » dans la balise INPUT et ce n'est pas lu par les lecteurs d'écran. Donc on devrait utiliser le Label de chacun de nos champs. On peut utiliser Float Label à la place, si on veut pouvoir le changer de place et le mettre, justement, à l'intérieur du champ.

Utiliser ARIA-DESCRIBEDBY pour les explications longues : si j'ai un formulaire complexe avec une longue explication, je vais utiliser la propriété aria-describedby pour que ce soit clair que c'est bien la description pour le formulaire.

Et on devrait garder les erreurs en contexte : une autre erreur, souvent, c'est qu'on va avoir un formulaire, puis si j'oublie d'écrire mon courriel ou bien si j'écris mon courriel en oubliant de mettre le point, par exemple, il y a certains formulaires qui vont me ramener dans le haut de la page où il y a tous les messages d'erreur. On ne devrait pas le faire comme ça. On devrait écrire les erreurs pour chacun des champs. Donc, si j'oublie mon courriel, l'erreur devrait être où j'écris mon courriel.

Les tableaux : on devrait utiliser les balises TH et l'attribut SCOPE pour les en-têtes. Je vois très rarement ces balises utilisées. En gros, ça nous permet de définir la colonne fait telle chose, telle autre colonne explique telle autre chose, etc. Ça permet vraiment de donner un contexte aux lignes et aux colonnes dans un tableau.

Utiliser THEAD, TBODY et TFOOTER : n'aident pas les lecteurs d'écran, mais permettent de garder les éléments en contexte, ce que je viens d'expliquer.

Autres notes sur les éléments HTML :

Les éléments cachés avec VISIBILITY:HIDDEN ou bien DISPLAY:NONE ne sont pas vus par les lecteurs d'écran. Utiliser le positionnement avec Z-INDEX à la place. Pour le z-index, un des trucs qu'on va utiliser souvent, le visibility:hidden, c'est pour les accordéons. Si j'ai un accordéon dans un site et que je clique sur la petite flèche, ça va cacher l'élément précédent et afficher le suivant. Souvent, on va utiliser la propriété visibility :hidden. Le problème avec ça, c'est que ce ne sera pas visible pour les lecteurs d'écran, donc, à la place, on devrait utiliser un z-index ou une quelconque autre méthode, mais on ne devrait pas cacher de contenu si c'est du contenu pertinent pour les gens qui ont des problèmes de vision.

Même chose pour les canevas : si on a des sites hyper-dynamiques avec un paquet d'animation, c'est très possible que ça utilise le canevas. On devrait s'assurer de référencer via JavaScript, ou avec ARIA pour s'assurer au moins qu'il y ait un texte qui explique aux gens qui ont un problème de vision ce qui est supposé se passer dans ce canevas.

Utiliser la balise BUTTON. Ne pas utiliser de balise DIV pour créer des

boutons. Encore une fois, c'est de s'assurer que les contenus soient en contexte, qu'on utilise un HTML sémantique. On ne devrait pas utiliser une boîte DIV, qui peut contenir n'importe quoi, en fait, alors qu'un bouton, le nom le dit, c'est un bouton, c'est un truc interactif.

On doit s'assurer d'avoir un bon niveau de contraste entre l'avant-plan et l'arrière-plan. Un titre bleu pâle sur un arrière-plan bleu foncé, il y a beaucoup de gens qui ont des problèmes de vision qui ne pourront pas le voir.

Ne pas mettre de contenus importants dans les scripts JavaScript. Même chose pour les lecteurs d'écran. Un script JavaScript normalement ne devrait pas avoir de texte. Ça devrait être disponible pour tout le monde, donc on s'assure de garder nos textes dans des balises HTML.

Ici, j'ai trois outils en ligne : le premier, c'est un site Web qui nous permet de vérifier, justement, le contraste de couleur entre l'avant-plan et l'arrière-plan. Ensuite, ce sont deux extensions qui permettent de vérifier le contraste de couleur entre l'avant-plan et l'arrière-plan et un paquet d'autres éléments, par exemple est-ce que les images ont bien toutes leurs propriétés ALT? J'ai mis Extension Firefox et Extension Chrome. Je n'utilise pas Safari – j'ai fait une recherche et je n'en ai pas trouvé. Je vous conseille d'utiliser un de ces deux navigateurs si vous devez vérifier l'accessibilité d'un site.

Les cas de figure : qu'est-ce qu'un site Web accessible, par exemple pour le spectre de l'autisme?

On doit s'assurer d'utiliser des couleurs simples, d'utiliser un langage clair et court, avoir des phrases courtes, des listes à points, des boutons descriptifs, des interfaces simples et consistantes. S'assurer que l'expérience utilisateur soit le plus simplifiée possible, le plus claire possible. C'est ce qui est le plus important.

Pour les lecteurs d'écran, donc pour les gens qui ont des problèmes de vision et les gens aveugles :

Bien décrire les images et sous-titrer les vidéos – je sais que sous-titrer les vidéos peut être problématique dans certains cas, mais si on veut s'assurer

que notre site est 100 % accessible, on doit sous-titrer les vidéos. On doit faire une mise en page linéaire et logique – on doit s'assurer que par exemple les formulaires, s'ils sont sur deux colonnes, que c'est facile de naviguer avec le Tab. Utiliser les balises sémantiques, navigation au clavier seulement et liens et titres descriptifs. On a vu ces éléments.

Troubles de la vision – ça peut être par exemple des gens qui sont daltoniens, pas seulement les gens qui sont aveugles :

Bon contraste de couleurs et bonne grosseur de police. Toute l'information utile sur la page (pas dans des images, vidéos ou PDF) – les contenus d'une page Web ne devraient jamais être dans un PDF; si vous avez le PDF, peut-être mettre le contenu du PDF en HTML et mettre un lien vers le PDF par la suite. Utiliser une combinaison logique de couleurs, formes et textes. Mise en page linéaire et logique. Garder les boutons et notifications en contexte – ne pas mettre les messages d'erreur du formulaire en haut, par exemple, ce que j'expliquais tantôt.

Accessibilité pour la dyslexie :

Utilisation de diagrammes et images pour supporter le texte. Aligner le texte à gauche et garder une mise en page consistante – s'amuser à faire des trucs hyper-complexes visuellement, ça n'aidera pas pour la dyslexie. Penser à créer du contenu sur d'autres médiums – si une personne a de la difficulté à lire, une vidéo, un podcast ou un enregistrement audio pourrait aider. Garder le contenu court, simple et clair et laisser les visiteurs changer les couleurs d'arrière-plan et d'avant-plan – si vous utilisez WordPress, il y a beaucoup de plug-ins qui permettent de laisser l'utilisateur modifier l'interface du site selon ses besoins et plusieurs sont gratuits.

Les gens qui ont des troubles moteurs ou physiques :

Créer des éléments cliquables faciles à cliquer – souvent, je vois des boutons qui sont collés sur un texte ou collés sur le bord de l'écran, très difficiles à aller chercher. Donner de l'espace aux éléments cliquables – même chose, par exemple pour quelqu'un qui utilise un téléphone mobile avec le pouce : s'il n'y a pas d'espace autour, ça va être difficile de le sélectionner. Design pour la navigation au clavier ou lecteur d'écran. Mobile First : on part de la version mobile et on va créer le site pour les

versions tablette et grands écrans. Minimiser les interactions, par exemple seulement le code postal au lieu de l'adresse complète – est-ce que vous avez réellement besoin d'avoir les coordonnées complètes d'une personne si c'est pour l'inscription à une infolettre, par exemple? Probablement pas. Donc, ne pas demander trop d'information à une personne si on n'en a pas besoin et essayer de minimiser les interactions. On a toujours dit qu'on devrait pouvoir accéder à tout le site en seulement trois clics. Dans ce cas-ci, c'est très important.

Pour les malentendants :

Un langage simple et clair – c'est aller droit au but. Sous-titres ou transcription pour les vidéos. Des mises en page linéaires et logiques. Diviser les contenus en sections – ça peut simplifier la lecture. Pour la prise de rendez-vous, permettre à l'usager de laisser une note (par exemple : demander la présence d'un interprète).

Pour les troubles d'anxiété :

Laisser assez de temps aux usagers pour compléter une action – pour certains sites gouvernementaux ou des sites où on doit remettre une information confidentielle, souvent il va y avoir un chronomètre sur un formulaire. Par exemple, on demande de terminer le formulaire en 20 minutes. On pourrait le laisser 60 minutes, à la place. On devrait laisser le temps aux usagers de compléter l'action facilement, sans être stressés par rapport au temps qui reste.

Expliquer ce qui se passera après avoir commis une action. Lorsqu'on a un formulaire avec un bouton « Envoyer », peut-être juste avant le bouton « Envoyer », dire : « Vous serez redirigé vers une page qui vous donnera la confirmation que nous avons bien reçu le formulaire. »

Mettre l'accent sur les informations importantes : les mettre en caractères gras ou en plus gros. Offrir le support nécessaire pour aider les usagers : si vous avez un formulaire hyper-complexe sur le site, peut-être mettre une vidéo d'une personne qui remplit le formulaire, par exemple. Laisser l'usager vérifier les réponses avant d'envoyer un formulaire. Peut-être que si la personne a cliqué sur « Envoyer », un pop-up qui dit : « Avez-vous bien vérifié telle ou telle section? », etc.

Ici, c'est plusieurs références pour le WCAG 2.0, le Trésor du gouvernement, qui était par rapport aux standards et aux normes de l'accessibilité Web. Si vous allez sur le site de quebec.ca/accessibilite, vous allez avoir l'information à propos du gouvernement du Québec et ce qu'ils ont fait pour s'assurer que leur site soit accessible.

a11yqc.org est un organisme québécois qui vient en aide aux entreprises pour s'assurer que leur site soit accessible.

ukhomeoffice – c'est une affiche qui parle de l'accessibilité. Les cas de figure que je viens d'expliquer viennent de ce poster.

Et le WAI-ARIA 1.0, etc.

C'est tout! C'est ce que j'avais à dire à propos de l'accessibilité Web au Québec.

Comme je l'expliquais au début de la présentation, ce n'est pas quelque chose d'obligatoire au Québec de rendre un site accessible. Par contre, ça devrait faire partie des plans de création d'un site Web. Ça devrait être une des premières choses qui est discutée.

Ce n'est pas difficile de rendre un site accessible. Ce n'est pas beaucoup plus de travail que de faire un site non accessible. Ce n'est pas ce qui va coûter le plus cher lorsqu'on fait le développement d'un site Internet.

Donc on se doit de faire un site accessible. Il y a tellement d'avantages à avoir un site accessible et très, très peu de désavantages. On se doit de faire des sites accessibles et de s'assurer que tous les gens puissent accéder à notre site.

C'est tout. Si je ne me trompe pas, il va y avoir une discussion planifiée avec OBORO. Je vais leur laisser l'opportunité d'en parler.

Merci beaucoup.

(Fin de la transcription)

Transcription : Marie Lauzon, trad. a. (Canada)

Pour citer cette conférence : Cédric Anderson, « L'accessibilité commence dans la planification : comment développer des sites Web accessibles? » (10 juin 2020). Communication présentée dans le cadre de la série *Interroger l'accès* d'OBORO et Spectrum Productions. Disponible en ligne : http://www.oboro.net/fr/activite/l-accessibilite-commence-dans-la-planification-comment-developper-des-sites-web-accessibles

OBORO

www.oboro.net

SPECTRUM PRODUCTIONS

www.productionsspectrum.com

Conseil des arts du Canada | Canada Council for the Arts

(Fin du document)