Accessibilité de la navigation

Principaux enjeux

- **Un site doit être entièrement navigable avec un clavier.**
  Débrancher la souris et tester.

- **Les mécanismes reliés à la navigation doivent être annoncés clairement.**
  Si un bouton a pour rôle de déployer ou cacher une navigation,
  un **texte descriptif de cette action** doit accompagner le bouton,
  être lisible et audible.

  Exemples de textes à ajouter :
  - bouton hamburger (action : Ouvrir le menu): *Menu*
  - bouton X (action : Fermer le menu): *Fermer*

- **La cible des hyperliens doit être claire et précise.**
  Ce qui se trouve à l'intérieur d'une balise <a> doit rendre explicite la cible de navigation.

Table des matières h1-h6

La table des matières générée par les titres h1-h6 doit représenter fidèlement la structure des contenus de la page Web car elle sert de menu dans les lecteurs vocaux tels que Jaws ou NVDA.

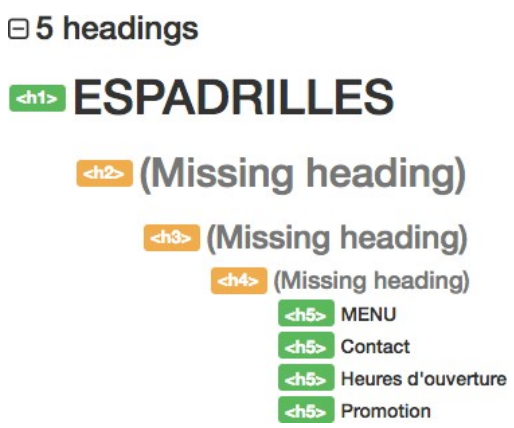
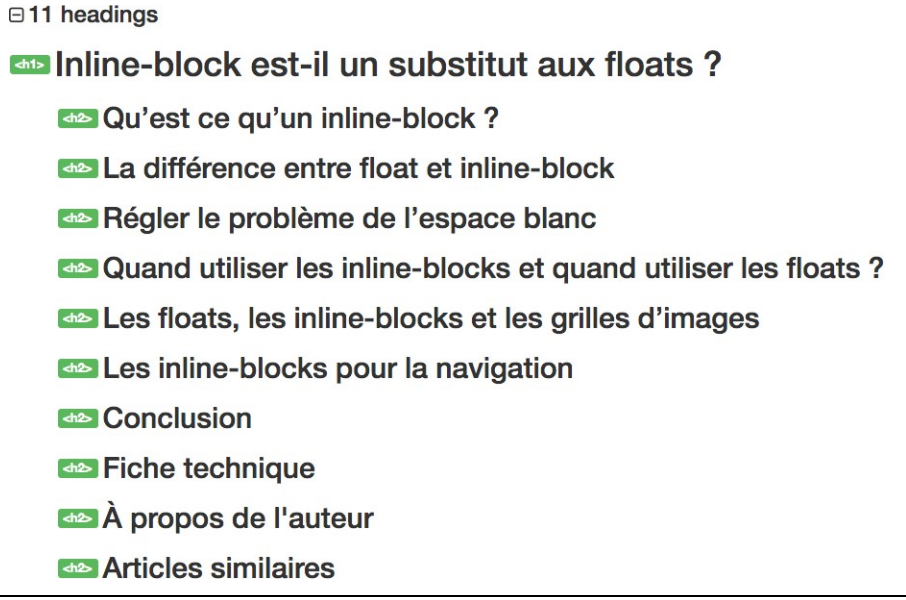
Vidéo : https://www.youtube.com/watch?v=QP1rzxMRap4

Vérifiez que la table des matières est représentative des contenus de la page Web en se servant de la **barre d'outils de développement** :

menu Informations > view document outline.

Pour faire cette vérification, vous devrez prendre la peine de lire les contenus, pour valider que les h1-h6 représente bien la hiérarchie de ces contenus.

Ce qui n'est pas spécifique aux contenus de la page et qui se trouve répété dans les autres pages (navigation, zone de recherche …) ne devrait pas figurer dans la table des matières.

Mauvaise utilisation des h1-h6	Hiérarchie correcte
	

Menu formé par les hyperliens dans le document

Les lecteurs vocaux offrent un menu constitué des libellés des hyperliens contenus dans le document Web.

Il est donc essentiel de s'assurer que le contenu des balises <a> sera représentatif de la cible de ces hyperliens…

Bogue si le texte des hyperliens est redondant
Vidéo : https://www.youtube.com/watch?v=QLfn-uABjuU

Solutions possibles pour des libellés d'hyperliens redondants

1. Design

Ne pas placer de texte « lire la suite » et placer le lien sur le titre de l'article.

2. Design et intégration

Ajouter une image dans la balise a et utiliser le alt pour compléter l'info de la cible de l'hyperlien

```
<a  href="titre-article.html">
    Lire la suite
    <img src="fleche.gif" alt="Titre de l'article" />
</a>
```

3. Intégration avec attribut aria-label[1]

```
<a  href="titre-article.html"
    aria-label="Lire la suite de Titre de l'article" >
    Lire la suite
</a>
```

[1] Cet attribut est documenté dans la norme ARIA (**Accessible Rich Internet Applications)**.

Menu de liens rapides

Dans le code html, il y a souvent un très grand nombre d'hyperliens de menus

- principaux,
- secondaires,
- contextuels,
- ou encore des liens publicitaires

AVANT d'arriver au vrai contenu de la page.

Or, pour une personne naviguant la page au clavier ou une personne utilisant un lecteur d'écran ou encore un outil de grossissement partiel de l'écran, cela exigera beaucoup de temps avant de comprendre de quoi il est question dans la page courante.

Vidéo : https://www.youtube.com/watch?v=XcqbyrsW7PQ (seulement la 1ère minute)

Bons exemples :

- WAI, https://www.w3.org/WAI/
- LinkedIN, https://www.linkedin.com/

Le menu de liens rapides propose une solution très simple à cela :

Tout de suite après l'ouverture de la balise **<body>**, on placera un hyperlien "Aller au contenu" qui pointera sur une ancre placée sur le premier **<h1>** ou le **<main>**.

On parle de **menu de liens rapides** (*skip links menu*) car lorsque c'est possible et pertinent, on peut ajouter à ce lien d'autres liens rapides de grande utilité pour les clientèles ayant des besoins particuliers, par exemple, un lien vers l'**outil de recherche**.

WEBAIM recommande de s'en tenir à un ou deux liens rapides.

(Source : http://webaim.org/techniques/skipnav/#multiple)

«Ça ne fait pas partie du design de mon site Web!»

Il est fort possible en effet que ce lien "Aller au contenu", inutile pour plusieurs usagers, soit considéré comme indésirable.

Le cacher… oui mais… il doit être **focussable** et **rendu visible au focus**.

C'est ici que les classes : **`screen-reader-only`** et **`focusable`** que nous avons ajoutées dans le fichier **_utilitaires.css** deviennent utiles en étant appliquées conjointement!

```
<body>
      <a href="#contenu" class="screen-reader-only focusable">
        Allez au contenu</a>
```

Menu des régions du document (*landmarks*)

Il est assez simple de mettre en place un autre mécanisme de navigation pour les lecteurs vocaux; en effet, ceux-ci sont capables de repérer **les régions du document** pour former un menu spécifique.

Vidéo : https://www.youtube.com/watch?v=TsIng1aDrIc

Les principales **régions du document** (en anglais ***landmarks***), sont :

- le bandeau d'entête,
- le contenu principal,
- la navigation,
- le pied de page,
- le ou les contenus complémentaires,
- les formulaires,
- la zone de recherche s'il y en a une.

Principales régions d'un document Web,
balises html pour les identifier et/ou rôle ARIA.

Légende
La balise et le rôle associé doivent être utilisés ensemble.

Description	Balise	Role (ARIA)
Bandeau d'entête	`header`	`banner`
Contenu principal	`main`	`main`
Navigation	`nav`	`navigation`
Pied de page	`footer`	`contentinfo`
Contenu complémentaire	`aside`	`complementary`
Formulaire	`form`	`form`
Zone de recherche	`form`	`Search`
Exemple : <main role="main">		

Un menu principal responsive et accessible

(voir le cadriciel)

Principes à mettre en œuvre

(1) Garder indépendants les styles et les comportements

Plutôt que de changer les styles des éléments directement en JavaScript, il vaut mieux prévoir chaque changement d'apparence en rédigeant une classe.

Exemple, on ajoute la classe **`nav--closed`** (navigation fermée) pour faire disparaître la liste en supprimant sa hauteur :

```
.js .nav__list {

    max-height: 100vh;
    …
}
```

```
.js .nav--closed .nav__list {

    max-height: 0;
    …
}
```

Les mécanismes contrôlés par JavaScript sont ajoutés par JavaScript.

Le html de base ne doit pas contenir les boutons qui sont actionnés par JavaScript.

Ainsi, si JavaScript est absent ou dysfonctionnel, il n'y a pas de boutons inutiles.

HTML de base

```
<nav class="nav nav--closed">

    <!-- C'est ici que le bouton hamburger sera ajouté par JavaScript -->
    <ul class="nav___list">
        <li class="nav___listItem">
            <a href="#" class="nav___link">Lien 1</a>
        </li>
        ...
    </ul>

</nav>
```

HTML après exécution de la méthode configurerNav du script _menu.js

```
<nav class="nav nav--closed">

    <button class="nav__control">
        <span class="nav__span">Menu</span>
    </button>

    <ul class="nav___list">
        <li class="nav___listItem">
            <a href="#" class="nav___link">Lien 1</a>
        </li>
        <li class="nav___listItem">
            <a href="#" class="nav__link">Lien 2</a>
        </li>
        ...
    </ul>

</nav>
```

Principes à mettre en œuvre

(2) Les contenus sont visibles par défaut

Assurer l'accessibilité universelle des contenus en développant d'abord la base pour les pires conditions possibles :
sans JavaScript, sans CSS, sans images.

Dans ces conditions,
le **menu principal** doit être présenté comme une liste d'hyperliens;
un **accordéon** doit être complètement déployé;
un **carrousel** d'images doit présenter toutes les images, etc.

Principes à mettre en œuvre

(3) Utiliser la classe conditionnelle "js"

Au début de notre JavaScript, on ajoute au `<body>` la classe "js".

```
let menu = {

  jsActif: document.body.classList.add("js"),

  …
```

Cela nous permettra d'écrire des sélecteurs conditionnels à la présence de JavaScript.

```
.js .nav__list {
    max-height: 100vh;
    overflow: hidden;
    transition: max-height 2s;
}
```

```
.js .nav--closed .nav__list {
    max-height: 0;
    transition: max-height 0.5s
}
```

Principes à mettre en œuvre

(4) Au besoin, utiliser la balise noscript pour présenter une solution alternative

S'il n'est pas possible de garder les contenus et les fonctionnalités disponibles sans JavaScript, alors, on peut offrir une alternative en se servant de la balise noscript (Référence : https://developer.mozilla.org/fr/docs/Web/HTML/Element/noscript).

La balise noscript se place habituellement dans le body du document, soit tout de suite au début ou en lieu et place ou doit apparaître l'alternative qu'elle propose.

Exemple d'utilisation de la balise `noscript`
lors de l'affichage de l'application **gmail** dans Chrome.