À propos de l'éditeur

Cet article est publié par le magazine Le train de 13h37.
Le train de 13h37 est un magazine en ligne francophone dédié à la conception Web qui publie régulièrement des articles exclusifs rédigés par des auteurs invités et rémunérés.

Nous avons fait le choix d'une publication gratuite sur notre site, sous licence Creative Commons BY-NC-SA et non financée par la publicité.

Nous éditons aussi des livres papiers et numériques, avec les Éditions En Voiture Simone.

Magazine édité par la société WAGON 42 SAS, au capital de 5 000€ - SIRET : 752 205 856 000 12
25 Impasse Ernest Feuillet - 84000 Avignon // 09.81.61.08.07 – contact@wagon42.fr

Tutoriel : créer un menu déroulant accessible en CSS

Définitions et usage

Un menu déroulant présente une liste de liens lorsque l'utilisateur survole ou clique sur un élément, et se trouve en général dans une barre de navigation.
Ce mode d'affichage nous vient tout droit des premières interfaces graphiques (le fameux *WIMP* : fenêtres, icônes, menus, pointeur) et sert à cacher les contenus secondaires jusqu'à ce qu'ils soient utiles.

Voici un code HTML basique pour un menu à deux niveaux :

```
<ul class="nav">
   <li class="nav-item"><a href="#">Accueil</a></li>
   <li class="nav-item">
      <a href="#">Catalogue</a>
      <ul class="nav sub-nav">
         <li class="sub-nav-item"><a href="#">Petits machins</a></li>
         <li class="sub-nav-item"><a href="#">Gros machins</a></li>
      </ul>
   </li>
   <li class="nav-item"><a href="#">A propos</a></li>
</ul>
```

Et voici comment le transformer en une liste horizontale dont le second niveau s'affiche verticalement.

```
/* Supprime les styles appliqués par défaut aux listes. S'applique également à la sous-liste */
.nav {
   list-style: none none;
   margin: 0;
   padding: 0;
   line-height: 1;
}
.nav a {
   display: block;
   padding:.5em;
   color: blue;
   background: white;
   text-decoration: none;
}
.nav a:focus,
.nav a:hover {
   color: black;
   background: #ffc;
   text-decoration: underline;
```

```
}
.nav-item {
   float: left; /* Pour que les liens s'affichent horizontalement */
   position: relative; /* Crée un contexte de positionnement pour les sous-listes */
}
.sub-nav {
   display: none; /* Masque la sous-liste */
   position: absolute; /* Evite que la dimension du conteneur ne change quand la sous-liste est affichée */
   left: 0; /* FIX IE7 : pour que le sous-menu s'aligne avec son conteneur */
   top: 2em; /* FIX IE7 : pour que le sous-menu s'aligne avec son conteneur, on le positionne explicitement en bas du conteneur */
   white-space: nowrap; /* Pour que le texte ne revienne pas à la ligne */
   background: white; /* FIX IE7 : évite que la sous-liste ne disparaisse au moment où le curseur arrive au-dessus */
   margin-top: -2px; /* Pour que le sous-menu recouvre son parent, facilitant le passage de la souris de l'un à l'autre */
}
.nav-item:hover.sub-nav {
   display: block; /* Affiche cette sous-liste au survol de son conteneur */
}
```

Voir la démo 1
Note : ce n'est qu'une manière d'obtenir ce type d'affichage parmi d'autres.

Cacher les niveaux inférieurs

La première technique qui vient à l'esprit pour cacher les éléments de niveau 2 (ou plus) consiste à leur appliquer `display: none`, et à restaurer `display: block` au survol de leur parent. Seulement voilà, en faisant ça c'est comme si les éléments n'existaient plus dans le document : il est impossible d'y accéder en tabulant, et les synthèses vocales les ignoreront royalement.
Et `visibility: hidden` me demanderez-vous ? Ce n'est pas mieux, comme l'explique très bien cet article sur WebAIM. On fait comment alors ?
L'article sus-cité indique comment masquer visuellement un élément, tout en le gardant accessible pour les aides techniques (synthèses vocales, etc.) et quand on navigue au clavier. Voici le code CSS à utiliser :

```
.invisibleMaisAccessible {
   position: absolute;
   left: -10000px;
   top: auto;
   width: 1px;
   height: 1px;
   overflow: hidden;
}
```

Pour que nos éléments réapparaissent quand on survole leur parent, nous n'avons plus qu'à redonner la valeur par défaut à ces différentes propriétés.

L'épreuve du ~~feu~~ clavier

En cachant les sous-listes ainsi, il nous suffit de survoler leur parent pour les révéler : les liens qu'elles contiennent sont alors accessibles en tabulant. Malheureusement rien ne se passe quand on navigue au clavier, car nous avons transformé les sous-listes en rectangles de `1px` par `1px` desquels rien ne peut sortir (du fait de l'instruction `overflow: hidden`).

Retour visuel

Ceux qui naviguent au clavier sont loin d'être tous aveugles, croyez-moi ! Il ne faut donc pas s'attendre à ce que l'intitulé des liens soit forcément lu par une aide technique.

```
/* Supprime les styles appliqués par défaut aux listes. S'applique également à la sous-liste */
.nav {
    list-style: none none;
    margin: 0;
    padding: 0;
    line-height: 1;
}
.nav a {
    display: block;
    padding:.5em;
    color: blue;
    background: white;
    text-decoration: none;
}
.nav a:focus,
.nav a:hover {
    color: black;
    background: #ffc;
    text-decoration: underline;
}
.nav-item {
    float: left; /* Pour que les liens s'affichent horizontalement */
    position: relative; /* Crée un contexte de positionnement pour les sous-listes */
}
.sub-nav {
    position: absolute; /* Evite que la dimension du conteneur ne change quand la sous-liste est affichée */
    left: 0; /* FIX IE7 : pour que le sous-menu s'aligne avec son conteneur */
    top: 2em; /* FIX IE7 : pour que le sous-menu s'aligne avec son conteneur */
    white-space: nowrap; /* Pour que le texte ne revienne pas à la ligne */
    background: white; /* FIX IE7 : évite que la sous-liste ne disparaisse au moment où le curseur arrive au-dessus */
    margin-top: -2px; /* Pour que le sous-menu recouvre son parent, facilitant le passage de la souris de l'un à l'autre */
}
/* Réduit cette boîte à un carré d'1px de côté, dont le texte est déporté loin à gauche */
```

```
.sub-nav-item {
   position: absolute;
   left: -10000px;
   top: auto;
   width: 1px;
   height: 1px;
   overflow: hidden;
   float: left; /* Fix WebKit : force la largeur des sous-navigations à s'adapter automatiquement, pour pouvoir l'atteindre à la souris */
}
.nav-item:hover.sub-nav-item {
   position: static;
   left: auto;
   width: auto;
   height: auto;
   overflow: visible;
}
```

Voir la démo 2

Dans l'exemple 2, les liens sont bien accessibles mais ils restent invisibles lorsqu'on y accède en tabulant. Ça n'est pas très pratique…

Nous devons donc rendre visible l'élément ayant le `focus`, mais comment ? N'ayant malheureusement pas (encore ?) de sélecteur d'élément parent en CSS (pour des raisons de performance), nous devons cacher les éléments plutôt que la liste elle-même, et rétablir les valeurs par défaut lorsque les liens reçoivent le `focus`.

Et le survol ?

Maintenant que nous cachons les éléments au lieu de la liste, il faut ajuster le code appliqué au survol du parent pour que la liste s'affiche correctement. On ne va quand même pas laisser les pauvres utilisateurs de souris derrière !

```
/* Supprime les styles appliqués par défaut aux listes. S'applique également à la sous-liste */
.nav {
   list-style: none none;
   margin: 0;
   padding: 0;
   line-height: 1;
}
.nav a {
   display: block;
   padding:.5em;
   color: blue;
   background: white;
   text-decoration: none;
}
.nav a:focus,
.nav a:hover {
```

```
    color: black;
    background: #ffc;
    text-decoration: underline;
}
.nav-item {
    float: left; /* Pour que les liens s'affichent horizontalement */
    position: relative; /* Crée un contexte de positionnement pour les sous-listes */
}
.sub-nav {
    position: absolute; /* Evite que la dimension du conteneur ne change quand la sous-liste est affichée */
    left: 0; /* FIX IE7 : pour que le sous-menu s'aligne avec son conteneur */
    top: 2em; /* FIX IE7 : pour que le sous-menu s'aligne avec son conteneur */
    white-space: nowrap; /* Pour que le texte ne revienne pas à la ligne */
    background: white; /* FIX IE7 : évite que la sous-liste ne disparaisse au moment où le curseur arrive au-dessus */
    margin-top: -2px; /* Pour que le sous-menu recouvre son parent, facilitant le passage de la souris de l'un à l'autre */
}
/* Réduit cette boîte à un carré d'1px de côté, dont le texte est déporté loin à gauche */
.sub-nav-item a {
    position: absolute;
    left: -10000px;
    top: auto;
    width: 1px;
    height: 1px;
    overflow: hidden;
    float: left; /* Fix WebKit : force la largeur des sous-navigations à s'adapter automatiquement, pour pouvoir l'atteindre à la souris */
}
.sub-nav-item a:focus,
.nav-item:hover.sub-nav-item a {
    position: static;
    left: auto;
    width: auto;
    height: auto;
    overflow: visible;
}
```

Voir la démo 3

Les liens de deuxième niveau s'affichent maintenant pour les utilisateurs naviguant au clavier, mais ils ne peuvent pas voir l'ensemble du menu d'un seul coup d'œil. Qu'à cela ne tienne, CSS3 à la rescousse !

Améliorons le retour

On peut maintenant tabuler parmi les liens de la sous-liste, mais on ne peut pas savoir combien elle contient d'éléments à l'avance. Modifions donc le code CSS pour afficher la sous-liste quand le parent reçoit le `focus`, ce qui est un bon palliatif.

```
/* Supprime les styles appliqués par défaut aux listes. S'applique également à la sous-liste */
.nav {
   list-style: none none;
   margin: 0;
   padding: 0;
   line-height: 1;
}
.nav a {
   display: block;
   padding:.5em;
   color: blue;
   background: white;
   text-decoration: none;
}
.nav a:focus,
.nav a:hover {
   color: black;
   background: #ffc;
   text-decoration: underline;
}
.nav-item {
   float: left; /* Pour que les liens s'affichent horizontalement */
   position: relative; /* Crée un contexte de positionnement pour les sous-listes */
}
.sub-nav {
   position: absolute; /* Evite que la dimension du conteneur ne change quand la sous-liste est affichée */
   white-space: nowrap; /* Pour que le texte ne revienne pas à la ligne */
   left: 0; /* FIX IE7 : pour que le sous-menu s'aligne avec son conteneur */
   top: 2em; /* FIX IE7 : pour que le sous-menu s'aligne avec son conteneur */
   white-space: nowrap; /* Pour que le texte ne revienne pas à la ligne */
   background: white; /* FIX IE7 : évite que la sous-liste ne disparaisse au moment où le curseur arrive au-dessus */
   margin-top: -2px; /* Pour que le sous-menu recouvre son parent, facilitant le passage de la souris de l'un à l'autre */
}
/* Réduit cette boîte à un carré d'1px de côté, dont le texte est déporté loin à gauche */
.sub-nav-item a {
   position: absolute;
   left: -10000px;
   top: auto;
   width: 1px;
```

```
    height: 1px;
    overflow: hidden;
    float: left; /* Fix WebKit : force la largeur des sous-navigations à s'adapter automatiquement, pour pouvoir l'atteindre à la souris */
}
.sub-nav-item a:focus,
.nav-item a:focus +.sub-nav a,
.nav-item:hover.sub-nav-item a {
    position: static;
    left: auto;
    width: auto;
    height: auto;
    overflow: visible;
}
```

Voir la démo 4

Emballé c'est pesé !

Voici donc le code complet pour un menu déroulant accessible garanti sans JS :

```
/* Supprime les styles appliqués par défaut aux listes. S'applique également à la sous-liste */
.nav {
    list-style: none none;
    margin: 0;
    padding: 0;
    line-height: 1;
}
.nav a {
    display: block;
    padding:.5em;
    color: blue;
    background: white;
    text-decoration: none;
}
.nav a:focus,
.nav a:hover {
    color: black;
    background: #ffc;
    text-decoration: underline;
}
.nav-item {
    float: left; /* Pour que les liens s'affichent horizontalement */
    position: relative; /* Crée un contexte de positionnement pour les sous-listes */
}
.sub-nav {
    position: absolute; /* Evite que la dimension du conteneur ne change quand la sous-liste est affichée */
```

```
    white-space: nowrap; /* Pour que le texte ne revienne pas à la ligne */
    left: 0; /* FIX IE7 : pour que le sous-menu s'aligne avec son conteneur */
    top: 2em; /* FIX IE7 : pour que le sous-menu s'aligne avec son conteneur */
    white-space: nowrap; /* Pour que le texte ne revienne pas à la ligne */
    background: white; /* FIX IE7 : évite que la sous-liste ne disparaisse au moment où le curseur
arrive au-dessus */
    margin-top: -2px; /* Pour que le sous-menu recouvre son parent, facilitant le passage de la souris
de l'un à l'autre */
}
/* Réduit cette boîte à un carré d'1px de côté, dont le texte est déporté loin à gauche */
.sub-nav-item a {
    position: absolute;
    left: -10000px;
    top: auto;
    width: 1px;
    height: 1px;
    overflow: hidden;
    float: left; /* Fix WebKit : force la largeur des sous-navigations à s'adapter automatiquement,
pour pouvoir l'atteindre à la souris */
}
.sub-nav-item a:focus,
.nav-item a:focus +.sub-nav a,
.nav-item:hover.sub-nav-item a {
    position: static;
    left: auto;
    width: auto;
    height: auto;
    overflow: visible;
}
@media screen and (max-width: 480px) {
.nav-item {
        float: none; /* Remet les éléments les uns au-dessus des autres */
    }
.sub-nav {
        position: static; /* Replace la sous-liste dans le flux du document */
        white-space: normal /* Permet au texte de revenir à la ligne normalement */
    }
.sub-nav-item a {
        display: block; /* Pour que la ligne entière soit cliquable */
        width: auto; /* Annule width: 1px */
        height: auto; /* Annule height: 1px */
        position: static; /* Annule position: absolute */
        padding-left: 1em; /* Annule le padding vertical et en rajoute à gauche pour indiquer le
sous-niveau */
        overflow: visible; /* Annule overflow: hidden */
        float: none;
```

```
    }
}
```

Voir la démo 5

Certains puristes trouvent que l'interactivité devrait être mise en place exclusivement via JavaScript, mais ce discours n'est plus tenu par grand monde. Personnellement, je trouve que 10 lignes de CSS sont plus lisibles et surtout plus robustes que le code JS qui serait nécessaire pour obtenir le même effet. De plus, elles évitent de devoir inclure un énième fichier JavaScript (et le framework qui va avec), avec le risque qu'il ne se charge pas, car bloqué par un proxy trop tâtillon (le fameux PALC) par exemple.

Si on tient à utiliser JavaScript, alors utilisons-le pour améliorer significativement l'utilisabilité du menu, en empêchant par exemple le sous-menu de se refermer instantanément quand on sort du parent. Cela évite aux utilisateurs de devoir déplacer leur souris verticalement puis horizontalement. Tous vos utilisateurs l'apprécieront, qu'ils aient les mains tremblantes, qu'ils soient secoués par leur environnement, ou simplement qu'ils déplacent leur souris rapidement. L'accessibilité bénéficie à tout le monde !

Le cas des méga-menus

Depuis quelques années, on voit apparaître sur les sites des sous-menus extra-larges contenant une palanquée de liens, voire même du texte, des illustrations et que sais-je encore. Suivant l'implémentation choisie, ceux qui se déplacent au clavier peuvent être forcés de traverser une jungle de liens impénétrable avant de pouvoir atteindre le contenu de la page. Les liens d'évitement sont totalement indispensables dans ce cas-là !
Sachez que les méga-menus mal pensés peuvent être de véritables catastrophes ergonomiques. À ce propos, cet article du Nielsen Norman Group détaille quelques erreurs de conception courantes et le meilleur moyen de les éviter.

La technique présentée ci-dessus peut être appliquée à un méga-menu, mais les contraintes qu'elle impose sur le HTML et le CSS peuvent être rédhibitoires : en l'absence de liens d'évitement, les utilisateurs qui naviguent au clavier vous maudiront de les forcer à tabuler parmi des centaines de liens avant d'arriver au contenu principal.
Si vous choisissez de ne pas permettre de tabuler parmi tous les liens du méga-menu, pensez à les rendre accessibles par d'autres moyens. Vous pouvez par exemple permettre de naviguer au clavier uniquement parmi le sous-menu de la section en cours, ou en doublant les liens dans un méga-pied de page (si cette mode n'est pas déjà passée à son tour).

J'ai beau préférer les solutions CSS, ici JavaScript est probablement bien plus adapté et ce d'autant plus qu'il est très certainement déjà utilisé pour afficher/masquer les sous-menus. Mais attention : Rappelez-vous que ce que vous modifiez en JavaScript doit également être accessible. Ne réinventez pas la roue, utilisez donc ce plugin pour méga-menu développé par l'équipe Accessibilité de chez Adobe et disponible sous licence Apache.

Encore une chose

(Comme disait feu Steve.) En l'absence de pointeur (souris ou *trackpad*), il n'y a pas moyen de déclencher l'ouverture du sous-menu. Sur un petit écran on peut rendre les liens de niveau 2 visibles et les empiler les uns sous les autres, mais cette solution n'est pas totalement satisfaisante car ces

liens occupent alors constamment une place précieuse et déséquilibrent le rapport navigation/contenu sur tablette.
Plusieurs techniques permettent d'afficher un menu à la demande de l'utilisateur, je vous laisse choisir celle qui convient le mieux à vos projets parmi cette liste.

Voilà, vous savez maintenant comment rendre un menu déroulant accessible au clavier sans devoir toucher au graphisme.
Vous n'avez plus d'excuse pour pénaliser les utilisateurs qui naviguent au clavier !

Magazine édité par la société WAGON 42 SAS, au capital de 5 000€ - SIRET : 752 205 856 000 12
25 Impasse Ernest Feuillet - 84000 Avignon // 09.81.61.08.07 – contact@wagon42.fr

À propos de l'auteur

Intégrateur passionné, Goulven Champenois jongle avec l'accessibilité, l'ergonomie, les performances et le mobile pour une expérience utilisateur optimale intégrant les dernières évolutions technologiques.
Formé en grande partie grâce aux sites OpenWeb et Pompage, il rejoint ce dernier en 2006.
Trouvant le temps trop long entre deux éditions de Paris Web, il participe à l'organisation de Sud Web depuis 2011.

Il propose désormais ses services en tant que consultant web.

- @goulvench
- <Goulven Champenois>