Mission 1

CREER UN MENU RESPONSIVE DEROULANT EN HTML ET EN CSS

Présentation de l'exercice

Dans cette mission 1, nous allons **réaliser un menu responsive déroulant seulement en HTML et en CSS**, sans l'aide d'aucun autre langage comme le JavaScript par exemple.

Afin de retirer le maximum de cet exercice, il est fortement conseillé de posséder de bonnes bases en HTML et particulièrement en CSS.

Dans son état « normal », notre menu va ressembler à cela :

Ensuite, on va vouloir que notre menu s'adapte en fonction de la taille de l'écran de vos visiteurs, afin d'offrir une meilleure expérience de navigation.

Voilà donc le type d'affichage qu'on va chercher à avoir sur Smartphones notamment :

Cet exercice va se découper en différentes phases :

1. La phase de réflexion ;
2. La création du menu en HTML ;
3. La création du menu en CSS ;
4. L'ajout des fonctionnalités responsives.

Etape n°1 : La réflexion

On commencera toujours nos développements par une phase de réflexion. En effet, il est essentiel de savoir ce que l'on veut et ce dont on va avoir besoin pour y parvenir avant de se lancer dans le développement à proprement parler.

Ceci est une bonne pratique qui produit généralement de bien meilleurs résultats, tant au niveau de la propreté du code que de l'ergonomie finale.

De plus, on s'aperçoit qu'on gagne finalement très souvent du temps sur le projet global en retardant la phase de développement.

Dans notre cas, nous voulons créer un menu déroulant et responsive. La grande contrainte va être de ne réaliser ce menu qu'avec les deux langages HTML et CSS.

Pour cela, on sait qu'on va commencer par utiliser des éléments de listes pour créer notre menu (éléments **ul** et **li**), ainsi que des éléments de type a pour créer les liens de notre menu.

Comme on veut un menu déroulant, c'est à dire un menu comportant des sous-onglets, on va également devoir imbriquer des listes les unes dans les autres.

On veut que les sous menus soient cachés par défaut puis affichés seulement lors du survol de la souris par un utilisateur.

Pour cela, nous allons utiliser judicieusement la propriété CSS display ainsi que la pseudo **classe :hover.**

On va également vouloir donner des styles différents à chaque sous menu. Pour cela, nous allons utiliser des sélecteurs CSS complexes.

Enfin, on veut que notre menu soit responsive, c'est-à-dire qu'il puisse s'adapter selon la taille de l'écran des visiteurs.

Cela n'est pas évident à faire seulement en HTML et en CSS et nous allons donc devoir « ruser » un peu. Pour cela, je vous propose une méthode à base de case à cocher invisible.

On est prêt pour commencer !

Etape n°2 : Création du squelette HTML de notre menu déroulant responsive

Nous allons commencer par créer le squelette HTML de notre menu.

En effet, avant de penser à la mise en forme, c'est-à-dire à la « couche de peinture », il faut bâtir des murs solides.

Nous allons créer des liens factices pour notre menu car l'objet n'est pas ici de perdre du temps à créer d'autres vraies pages.

Pour cela, nous placerons un dièse (« # ») en valeur de l'attribut **href** de nos liens.

Nous n'aurons donc qu'une seule page **HTML** à créer, que l'on va appeler **menu.html.**

Voici le code complet pour cette page :

```html
<!DOCTYPE html>
<html>
    <head>
        <title>Menu responsive déroulant HTML / CSS</title>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <link rel="stylesheet" href="menu-deroulant-responsive.css">
    </head>
    <body>
        <nav>
            <label for="menu-mobile" class="menu-mobile">Menu</label>
            <input type="checkbox" id="menu-mobile" role="button">
            <ul>
                <li class="menu-html"><a href="#">HTML</a>
                    <ul class="submenu">
                        <li><a href="#">Cours HTML et CSS</a></li>
                        <li><a href="#">Eléments HTML</a></li>
                        <li><a href="#">Attributs HTML</a></li>
                        <li><a href="#">Exercices</a></li>
                    </ul>
                </li>
                <li class="menu-css"><a href="#">CSS</a>
                    <ul class="submenu">
                        <li><a href="#">Cours HTML et CSS</a></li>
                        <li><a href="#">Propriétés CSS</a></li>
                        <li><a href="#">Sélecteurs CSS</a></li>
                        <li><a href="#">Exercices</a></li>
                    </ul>
                </li>
                <li class="menu-js"><a href="#">JavaScript</a>
                    <ul class="submenu">
                        <li><a href="#">Cours JavaScript</a></li>
                        <li><a href="#">Fonctions JavaScript</a></li>
                        <li><a href="#">Le DOM</a></li>
                        <li><a href="#">Exercices</a></li>
                    </ul>
                </li>
                <li class="menu-contact"><a href="#">Contact</a></li>
            </ul>
        </nav>
    </body>
</html>
```

Je vais vous demander pour le moment de ne pas regarder les lignes contenant le meta name="viewport", le label et l'input. Nous expliquerons ces trois lignes de code plus tard lorsqu'on parlera du responsive.

Dans cette page, on commence par lier un fichier CSS à notre fichier HTML avec l'élément **link**. Nous allons créer et travailler sur ce fichier CSS par la suite.

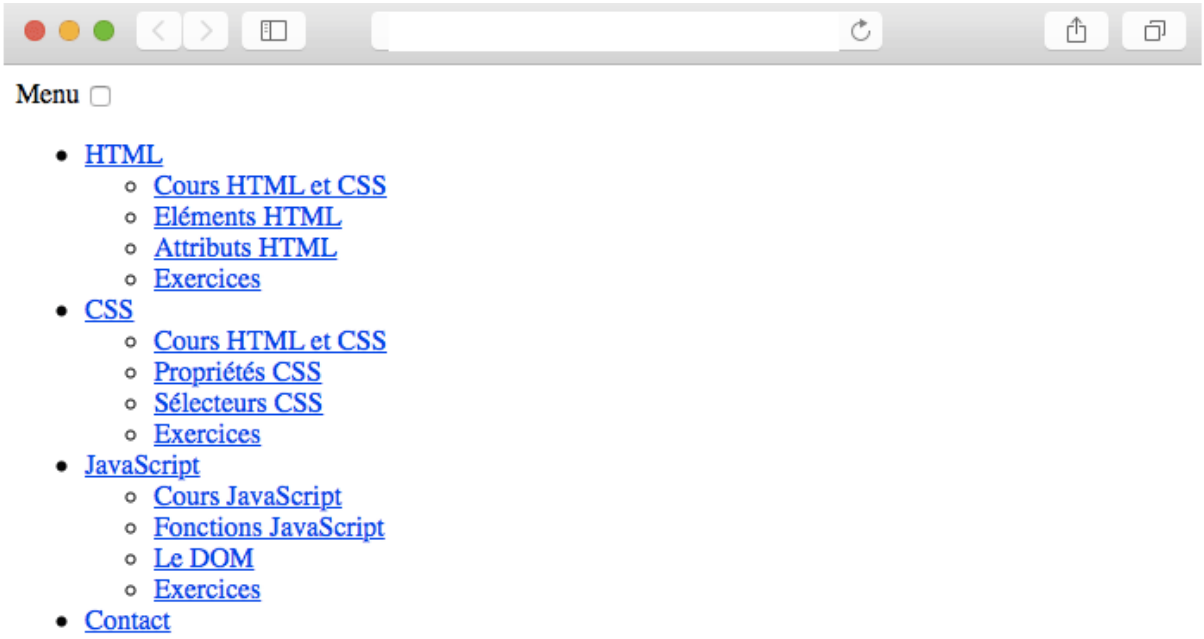
Ensuite, on encadre tout notre menu dans l'élément structurant **nav** qui sert à indiquer le menu de navigation principal de notre page.

Pour créer les onglets de notre menu, on utilise une liste ul et des éléments li pour représenter chaque onglet. Comme je vous l'ai précisé ci-dessus, on crée des liens factices en donnant la valeur « # » aux différents attributs **href**.

Enfin, on imbrique une nouvelle liste à l'intérieur de chaque élément de notre liste principale afin de créer nos sous onglets, sauf pour l'onglet « contact » qui ne possèdera pas de sous onglet.

Voilà tout pour la partie HTML de notre menu. Jusque là, rien ne devrait vous surprendre.

Pour le moment, notre menu ressemble à cela :



Comme vous pouvez le constater, il nous reste un gros travail de mise en forme à faire en CSS que nous allons attaquer dès maintenant.

Etape n°3 : Mise en forme CSS de notre menu HTML

La première chose à faire va être de créer notre fameux fichier menu-deroulant-responsive.css. Placez ce fichier dans le même dossier que notre fichier HTML pour plus de simplicité.

Ensuite, nous allons commencer par effectuer un reset CSS des marges intérieures et extérieures de notre élément body.

En effet, certains navigateurs attribuent par défaut des marges d'une certaine taille à cet élément. Comme on veut pouvoir travailler tous sur les mêmes bases, et avoir un affichage final optimal, ce reset est nécessaire.

On va également en profiter pour définir la police de notre page. On choisi dans le classique du **Verdana**.

```
body{
    font-family: Verdana, Calibri, sans-serif;
    margin: 0px;
    padding: 0px;
}
```

On va effectuer la même opération pour notre élément de liste principal (**ul**) qui va avoir le même souci :

```
/*On utilise ">" pour ne cibler que la liste ul qui
 *est un enfant direct de nav*/
nav > ul{
    margin: 0px;
    padding: 0px;
}
```

Comme précisé dans le code, on utilise le signe > pour ne cibler que la liste **ul** enfant direct de **nav**.

Ensuite, nous allons enlever les puces affichées automatiquement par le navigateur devant nos éléments de liste et nous allons placer tous les éléments de notre liste principale sur la même ligne.

Nous voulons que ce comportement ne s'applique qu'à nos onglets et non pas à nos sous onglets qui resteront eux en colonne. Pour cela, on utilise à nouveau le signe > pour cibler les éléments de notre liste principale uniquement.

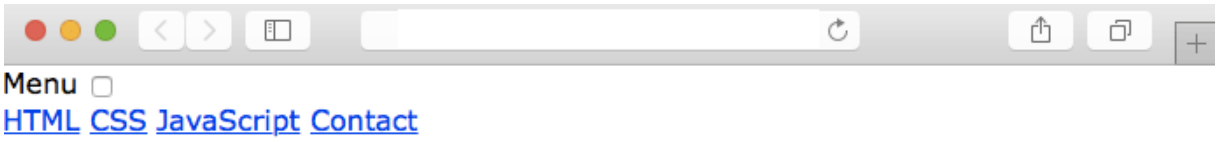
```
/*On enlève les puces devant tous les éléments li
 *appartenant à notre élément nav*/
nav li{
    list-style-type: none;
}

/*On affiche seulement les éléments li correspondant à
 *nos onglets de menu en ligne (côte à côte)*/
nav > ul > li{
    float: left;
}
```

Les sous onglets de notre menu ne doivent apparaître que lorsqu'on survole l'onglet auquel ils appartiennent avec la souris et doivent rester invisibles le reste du temps.

Pour le moment, nous allons déjà nous contenter de les cacher grâce à la propriété CSS display.

```
.submenu{
    display: none;
}
```

Voilà donc le rendu actuel :



Nous allons également en profiter pour cacher notre case à cocher ainsi que le mot « menu » situés au dessus de notre menu :

```css
/*On rend notre case à cocher invisible*/
nav input[type=checkbox]{
    display: none;
}
```

```css
/*On cache le bouton affichant le menu mobile*/
.menu-mobile{
        display: none;
}
```

Maintenant que l'on y voit un peu plus clair, nous allons véritablement pouvoir mettre en forme les éléments de notre menu.

Tout d'abord, on va commencer par étirer notre menu de façon à ce qu'il prenne toute la largeur disponible et lui donner une couleur de fond.

```css
nav{
    width: 100%;
    background-color: #424558;
}
```

Nous allons ensuite donner attribuer une position : relative aux éléments de notre menu principal car cela nous sera utile plus tard pour bien positionner les sous onglets.

```css
/*On affiche seulement les éléments li correspondant à
 *nos onglets de menu en ligne (côte à côte)*/
nav > ul > li{
    float: left;
    position: relative;
}
```

A ce niveau, cependant, nous allons être confronté à une difficulté en CSS.

En effet, vous devez savoir que si un élément parent non flottant possède des enfants flottants, le CSS va supprimer la hauteur du parent et celui ne va donc plus s'afficher.

Ici, la solution la plus évidente serait d'appliquer un overflow:hidden à notre élément parent afin de forcer le CSS à lui rendre sa hauteur.

Cependant, nous n'allons pas pouvoir procéder de la sorte dans le cas présent car utiliser un **overflow:hidden** rendrait totalement invisible nos sous onglets par la suite.

La solution va donc être d'utiliser ce qu'on appelle un « hack » CSS de type clearfix, très connu par les développeurs.

Pour cela, nous allons ajouter un contenu nul à la fin de notre liste principale, changer le type de display et appliquer un clear.

```
/*Un "hack" CSS (clearfix) très connu*/
nav > ul::after{
    content: "";
    display: table;
    clear: both;
}
```

Cette partie était certainement la plus complexe de l'exercice car il fallait bien comprendre comment le CSS fonctionne et car il fallait connaître le clearfix.

Maintenant que le problème est réglé, nous allons nous attaquer à la mise en forme de nos liens de menu.

Nous allons commencer par changer le type de display en inline-block de nos éléments a et supprimer le trait de soulignement qui apparait automatiquement avec ceux-ci.

Nous allons également attribuer une marge intérieure aux liens de nos onglets afin d'étendre la taille des onglets tout en faisant en sorte que les onglets restent cliquables dans leur ensemble.

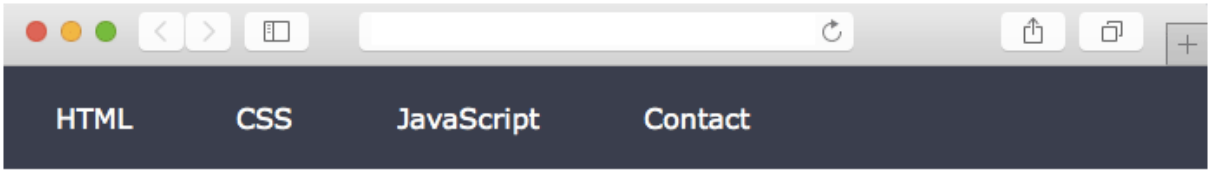
On va encore changer la couleur de nos liens de menu afin que ceux-ci soient bien visibles.

```css
nav a{
    display: inline-block;
    text-decoration: none;
}

nav > ul > li > a{
    padding: 20px 30px;
    color: #FFF;
}
```

Faites bien attention à bien cibler les éléments et, si possible, essayez de cibler les éléments le plus précisément possible.

En effet, ici, je n'ai pas besoin à priori de tous les signes >. Seulement, je les ai tous mis pour éviter des problèmes futurs si par exemple je modifie ou j'enrichis mon menu.

Nous en avons donc fini avec le premier niveau de notre menu. Voilà le résultat pour le moment :



Passons maintenant à la mise en forme de nos sous onglets.

Pour rappel, on veut que ces sous onglets ne s'affichent que lorsqu'un utilisateur passe sa souris sur l'onglet auquel appartiennent les sous onglets.

On va donc utiliser une pseudo classe :hover sur nos onglets.

On veut également que chaque sous onglet s'affiche exactement sous son onglet parent. Pour cela, rappelez vous que nous avons déjà utilisé position:relative pour nos onglets en CSS.

Nous n'avons donc plus qu'à utiliser un position:absolute sur nos sous menus et à les placer avec les propriétés top et left.

Nous allons également devoir reset le padding de nos sous menus afin de ne pas avoir de problèmes d'affichage à l'intérieur de ceux-ci.

```css
nav li:hover .submenu{
    display: inline-block;
    position: absolute;
    top: 100%;
    left: 0px;
    padding: 0px;
    z-index: 100000;
}
```

Comme vous pouvez le constater, j'ai également ici ajouté un z-index élevé. Cela est nécessaire pour que les sous menus s'affichent bien par dessus le reste du contenu de la page.

Pour améliorer le rendu de nos sous onglets, nous allons ajouter une bordure fine à chacun de nos sous éléments.

```css
.submenu li{
    border-bottom: 1px solid #CCC;
}
```

Nous allons également mettre en forme les liens de nos sous menus en leur ajoutant un padding, changeant la taille du texte et sa couleur et en définissant une largeur fixe pour ceux-ci.

```css
.submenu li a{
    padding: 15px 30px;
    font-size: 13px;
    color:#222538;
    width: 270px;
}
```

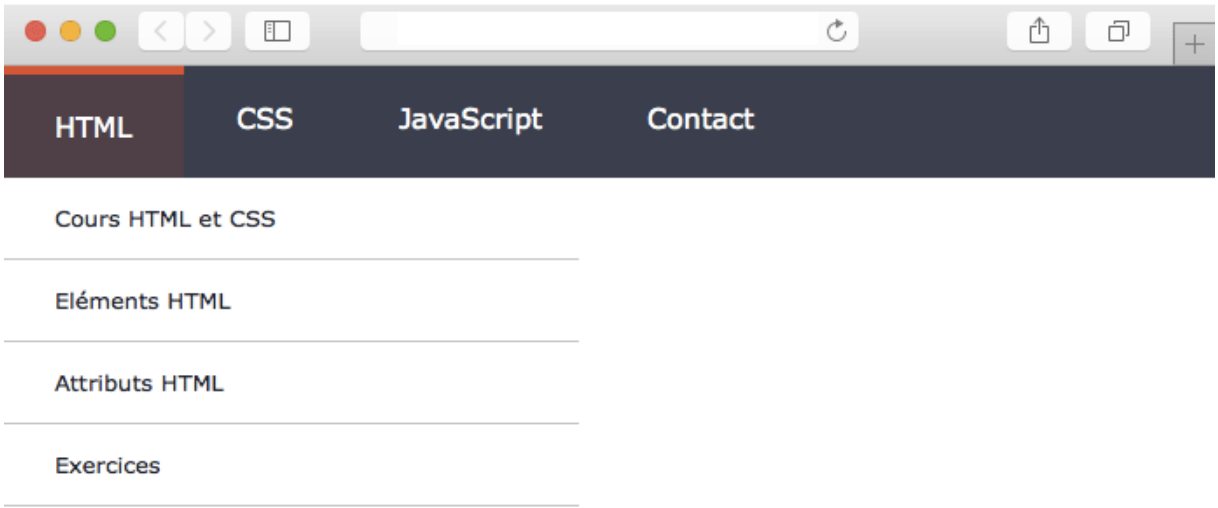
Il ne nous reste alors plus qu'à effectuer un dernier travail d'habillage pour nos onglets et sous menus en leur ajoutant notamment de la couleur.

Je vous propose pour cela de commencer par ajouter une bordure haute épaisse d'une couleur différente pour chacun des éléments de notre menu principal. Cette bordure n'apparaîtra que lors du survol de l'élément en question.

Lorsque l'élément est survolé, on va également lui attribuer une couleur de fond en semi transparence afin que le contenu de l'élément reste bien lisible.

```css
.menu-html:hover{
    border-top: 5px solid #e44d26;
    background-color: RGBa(228, 77, 38, 0.15);
}

.menu-css:hover{
    border-top: 5px solid #0070bb;
    background-color: RGBa(000, 112, 192, 0.15);
}

.menu-js:hover{
    border-top: 5px solid #f1dc4f;
    background-color: RGBa(241, 211, 79, 0.15);
}

.menu-contact:hover{
    border-top: 5px solid #BBB;
    background-color: RGBa(220, 220, 220, 0.15);
}
```

Le problème ici est que l'ajout de la bordure va créer un décalage du texte de nos éléments d'une hauteur équivalente à sa taille.



Nous allons remédier à cela en modifiant le padding de nos éléments seulement lorsque l'élément est survolé. Plus précisément, nous allons soustraire de la valeur du padding-top la taille de la bordure créée.

```css
nav > ul > li:hover a{
    padding: 15px 30px 20px 30px;
}
```

Il ne nous reste donc plus qu'à habiller nos sous onglets en leur ajoutant de la couleur.

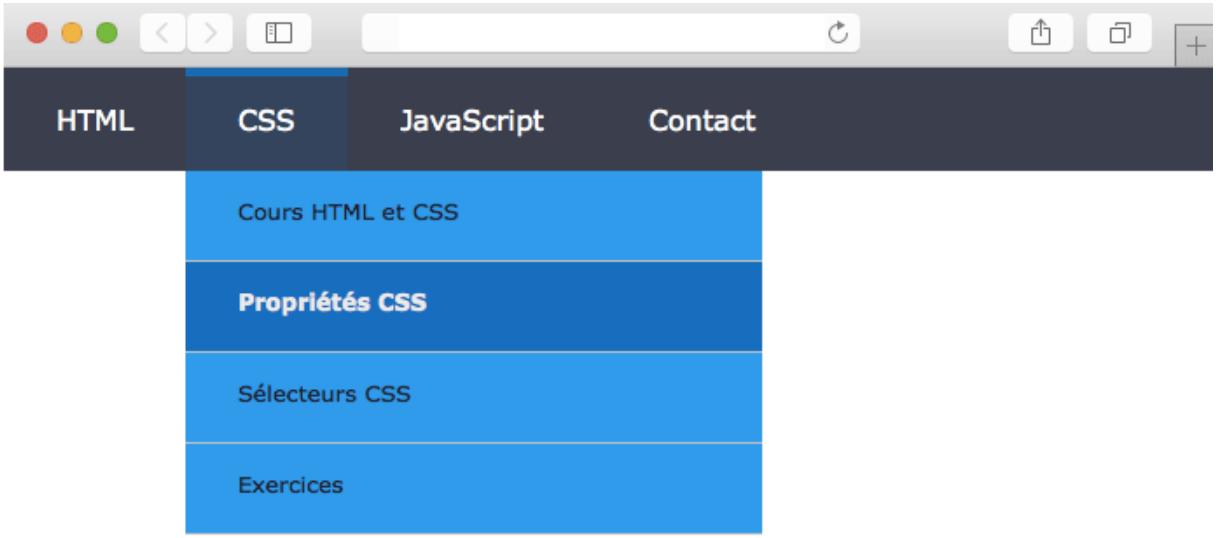
Pour cela, nous allons ajouter deux nuances de couleur à chacun de nos sous éléments, selon que la souris soit au dessus du sous élément ou non.

De même, nous allons changer la couleur du texte si la souris est au dessus ainsi que son poids.

Voici donc ce que je vous propose :

```css
.menu-html .submenu{
    background-color: RGB(230, 100, 40);
}
.menu-css .submenu{
    background-color: RGB(000, 160, 240);
}
.menu-js .submenu{
    background-color: RGB(251, 216, 99);
}

.submenu li:hover a{
    color:#EEE;
    font-weight: bold;
}
.menu-html .submenu li:hover{
    background-color: RGB(210, 77, 60);
}
.menu-css .submenu li:hover{
    background-color: RGB(000, 115, 200);
}
.menu-js .submenu li:hover{
    background-color: RGB(200, 165, 75);
}
```

Et voilà donc le résultat final pour notre menu :



Il ne nous reste plus qu'à nous occuper du côté responsive !

Etape n°4 : La gestion du responsive pour notre menu

Commençons par retourner sur notre page HTML afin d'expliquer les trois lignes dont nous ne nous sommes pas occupés jusqu'à présent.

```
<!DOCTYPE html>
<html>
    <head>
        <title>Menu responsive déroulant HTML / CSS</title>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <link rel="stylesheet" href="menu-deroulant-responsive.css">
    </head>
    <body>
        <nav>
            <label for="menu-mobile" class="menu-mobile">Menu</label>
            <input type="checkbox" id="menu-mobile" role="button">
            <ul>
                <li class="menu-html"><a href="#">HTML</a>
```

Le meta name="viewport", tout d'abord, va nous servir à faire une première « mise à l'échelle » de notre page web par rapport à la taille de l'écran du visiteur.

Cela va permettre entre autres de faire en sorte qu'une image ou qu'un texte se redimensionne automatiquement pour occuper l'espace correspondant à un plus petit écran.

Cela est un bon début pour créer une page responsive mais n'est cependant pas suffisant.

Pour ajouter une touche responsive à notre menu, il va nous falloir ruser un peu.

En effet, il n'est pas évident du tout de proposer une version responsive d'un menu en utilisant seulement du HTML et du CSS.

La solution que je vous propose est la suivante : nous allons créer une case à cocher en HTML avec l'élément input.

L'idée est d'afficher une autre version du menu en CSS si la case a été cochée.

Evidemment, nous voulons que cette case puisse n'être cochée que pour certaines tailles d'écran. Nous allons donc cacher l'élémentinput en CSS pour tous les autres écrans.

Nous avons déjà fait cela dans la partie précédente en appliquant un display:none à nos éléments label et input si vous vous rappelez bien.

```
.menu-mobile{
        display: none;
}

/*On rend notre case à cocher invisible*/
nav input[type=checkbox]{
    display: none;
}
```

Nous allons maintenant utiliser les media queries et la règles CSS @ pour créer le responsive de notre menu.

Nous allons vouloir afficher une version différente du menu pour tous les écrans d'une taille inférieure à 680px.

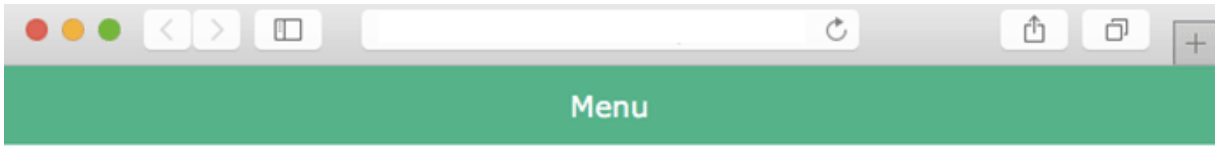
Nous allons déjà commencer par afficher notre label pour ces écrans et styliser celui-ci pour un meilleur affichage sur mobile.

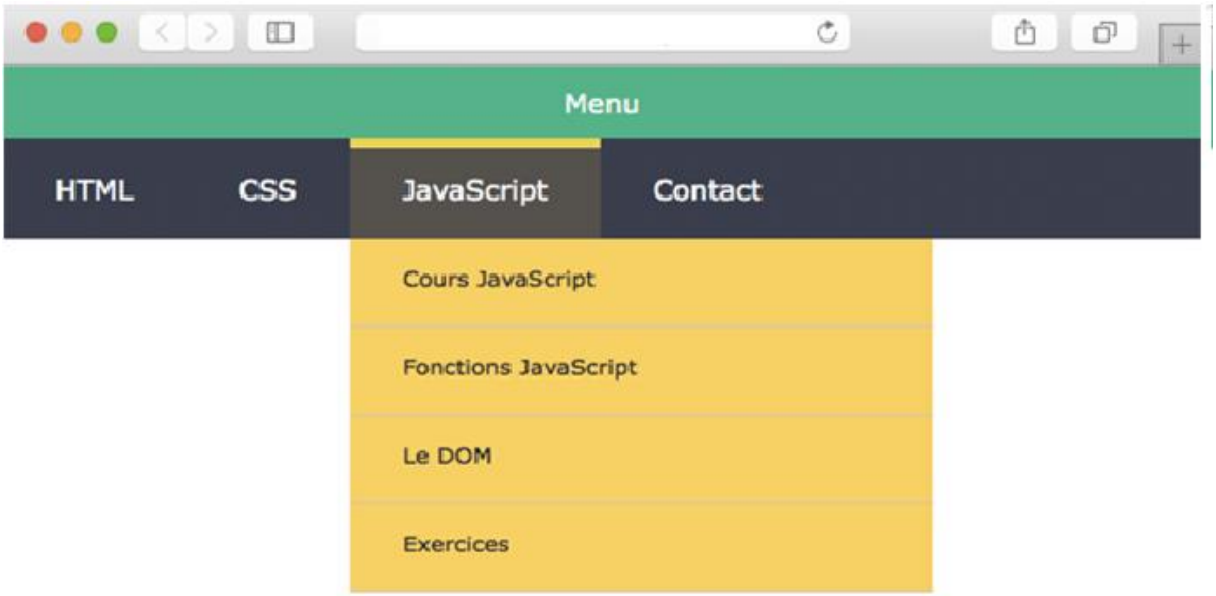
Nous allons également cacher tout le contenu du menu tant que l'utilisateur n'aura pas cliqué sur la case menu, c'est-à-dire n'aura pas coché la case.

Dès que la case est cochée, en revanche, le menu doit s'afficher.

```
/*Lorsque la case est cochée (dès qu'un utilisateur appuie sur
 *notre bouton), le menu entier s'affiche*/
nav input[type=checkbox]:checked ~ ul{
    display: block;
}

/*On modifie le style de notre menu pour son affichage responsive*/

@media screen and (max-width : 680px){
    .menu-mobile {
        display:block;
        color: #fff;
        background-color: rgba(29, 197, 151, 0.97);
        text-align: center;
        padding: 12px 0;
    }
    nav ul{
        display: none;
    }
}
```

Voilà à quoi ressemble notre menu responsive pour le moment, sans la case cochée et avec la case cochée.





Si vous travaillez sur ordinateur, pensez bien évidemment à redimensionner votre fenêtre pour que celle-ci ait une taille inférieure à 680px afin de voir votre menu responsive.

Nous allons maintenant modifier l'affichage de nos onglets et sous onglets pour optimiser l'affichage sur mobile.

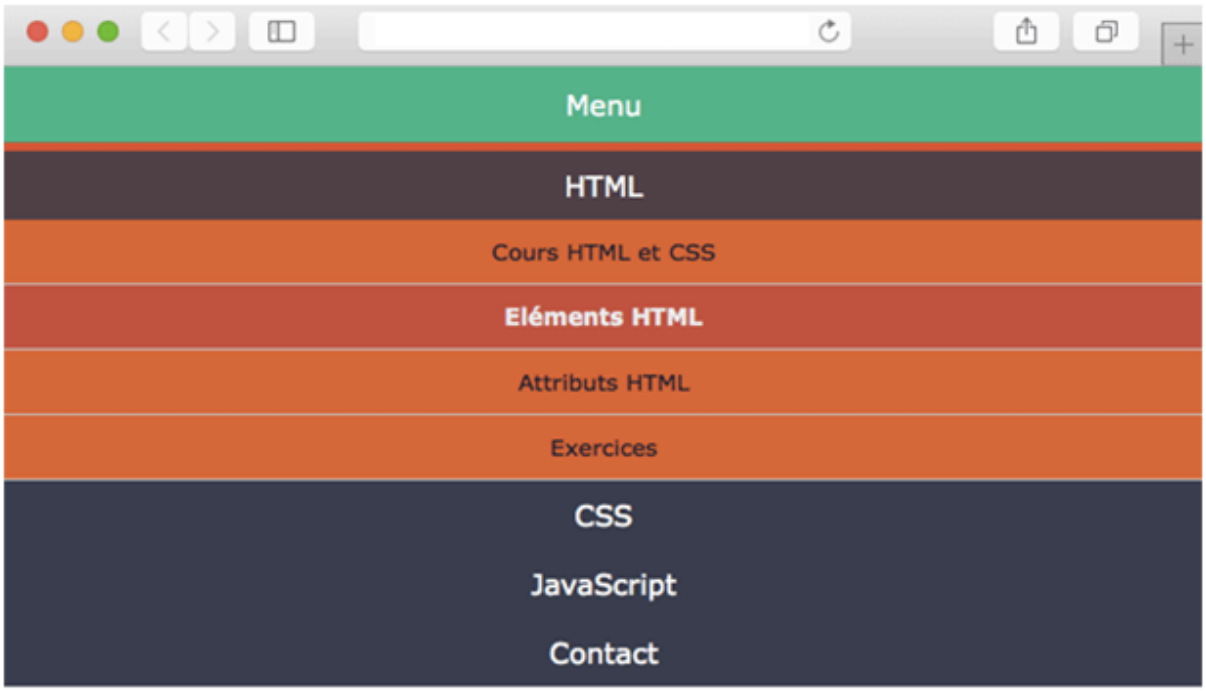
Pour cela, on va faire très simple et conserver la majorité de ce que nous avons fait dans les parties précédentes. Nous allons simplement étendre la taille des éléments de liste ainsi que celle des liens à 100%.

Nous allons également changer le display ainsi que le type de positionnement de nos sous menus afin que ceux-ci s'affichent bien sous les éléments de menu et prennent également tout l'espace disponible.

Finalement, on va ajouter un léger **padding** sur nos différents liens pour espacer un peu nos contenus.

```css
/*On modifie le style de notre menu pour son affichage responsive*/

@media screen and (max-width : 680px){
    .menu-mobile {
        display:block;
        color: #fff;
        background-color: rgba(29, 197, 151, 0.97);
        text-align: center;
        padding: 12px 0;
    }
    nav ul{
        display: none;
    }
}
    nav ul li, nav ul li a {
        width: 100%;
        text-align: center;
    }
    nav ul li a, nav ul li:hover a{
        padding: 10px 0px 10px 0px;
    }
    nav li:hover .submenu{
        display: block;
        position: static;
    }
}
```

Et voilà le résultat auquel nous parvenons finalement :



Conclusion : un menu responsive en HTML et en CSS

Cet exercice illustre bien selon moi toute la puissance du couple HTML / CSS ainsi que les possibilités que nous offrent ces deux langages.

Nous venons donc de créer un menu déroulant responsive parfaitement fonctionnel en HTML5 et en CSS3.

Vous pouvez évidemment par la suite améliorer ce menu ou lui appliquer simplement d'autres styles pour qu'il soit parfaitement adapté à vos besoins.

Notez cependant que l'on a dû un petit peu ruser afin de ne pas avoir recours à d'autres langages comme le JavaScript pour créer ce menu.

N'hésitez pas par ailleurs à aller apprendre le JavaScript si ce n'est déjà fait : ce langage va nous permettre d'étendre encore davantage les possibilités du HTML et du CSS !