M2202 – Algorithmique
TD 3 : Interactivité utilisateur et menus (suite)

Aperçu des méthodes de gestion de l'interactivité :

Action	Fonction
Clic	`click()`
Double-clic	`dblclick()`
Passage de la souris	`hover()`
Rentrer dans un élément	`mouseenter()`
Quitter un élément	`mouseleave()`
Presser un bouton de la souris	`mousedown()`
Relâcher un bouton de la souris	`mouseup()`
Scroller (utiliser la roulette)	`scroll()`

La méthode on()

Cette méthode permet de gérer l'ensemble des évènements (les méthodes présentées dans l'aperçu ci-dessus n'en sont que des raccourcis) en une fois. Elle accepte comme paramètre supplémentaire le nom de l'évènement :

```
$('button').on('click', function(){//le nom de l'évènement est passé sous forme de chaine
    alert('Ce code fonctionne !');
});
```

Il est possible de déclarer plusieurs évènements dans une seule méthode **on()** :

```
$('button').on({
    click : function(){//pas de quote cette fois !!
        alert('Vous avez cliqué !');
    },
    mouseup : function(){
        alert('Vous avez relâché le clic !');
    }
});
```

On gère dans cet exemple à la fois l'évènement **click** et **mouseup** pour tous les éléments **button**. Notez que **click** et **mouseup** sont dans ce cas des objets séparés (d'où la virgule) passés à la méthode **on()** dans un objet.

NB : jQuery propose la méthode **off()**, qui, à l'inverse de **on ()**, désactive les évènements pour les éléments ciblés.

Déclencher virtuellement un événement

jQuery permet de simuler le déclenchement d'évènements grâce à une simple méthode : **trigger()** ! Il suffit de passer le type de l'évènement en tant que paramètre.

```
$('p').click(function(){
    alert('Cliqué !');
});

$('p').trigger('click'); // déclenche l'action click pour tous les éléments p
```

1. **Exercice 1** : habillage d'une liste de liens

Il s'agit d'animer les liens contenus dans une liste non ordonnée. Dans un premier temps, nous nous contenterons de modifier la couleur des textes de ces liens au survol (de bleu foncé à violet). Nous utiliserons les objets **mouseenter** et **mouseleave** passés à la méthode **on()** pour effectuer ces modifications.
Le code sera donc le suivant :

```
<script src='http://ajax.googleapis.com/ajax/libs/jquery/1.10.2/jquery.min.js'>
</script>
<script>
    $(document).ready(function(){
        $('#navigation a').on({
            mouseenter :function(){$(this).css('color','purple');},//attention : virgule
            mouseleave :function(){$(this).css('color','darkblue');}
        });
    });
</script>
```

Comme on peut le constater d'après la sélection **('#navigation a')**, les éléments **a** et **li** seront placés dans un **conteneur** (**div**) dont l'attribut **id** aura pour valeur **navigation**.

Créez et enregistrez les pages html et css suivantes :

mapagejq5.html :

```
<!DOCTYPE html>
<html>
<head>
     <meta charset='utf-8'>
    <link rel='stylesheet' href='mapagejq5.css'>
</head>
<body>
    <div id='navigation'>
        <ul>
            <li><a href='#'>Lien 1</a></li>
            <li><a href='#'>Lien 2</a></li>
            <li><a href='#'>Lien 3</a></li>
            <li><a href='#'>Lien 4</a></li>
        </ul>
    </div>
</body>
</html>
```

mapagejq5.css :

```
@charset "utf-8";
/* CSS Document */

#navigation{
     width:10%;
}

#navigation ul {
     list-style-type: none;
     margin: 4px;
     font-size: 16px;
     font-weight: bolder;
```

```
     font-family:Arial, Helvetica, sans-serif;
}

#navigation li a{
     text-decoration:none;
     color:darkblue;
}

#navigation li{
     background-color:lightgrey;
}
//Fin css
```

Une fois les deux pages réalisées, insérez le code permettant de gérer les survol des liens. Enregistrez et vérifiez le bon fonctionnement du dispositif.

Selon le même principe, modifiez le code qui permettant de modifier la couleur d'arrière-plan des éléments **li** (bleu clair au survol). Plusieurs méthodes d'implémentation sont possibles, l'une est simple et consiste à modifier une copie du code existant, l'autre consiste à seulement modifier la fonction existante en prenant en compte le fait que les éléments **a** sont enfants direct des éléments **li**.

2. **Exercice 2** : animation des éléments de listes.

Il s'agit d'une fonctionnalité essentielle de ce qui constitue les animations de type 'accordéon'. Les éléments (liste, section, etc.) sont révélés de façon progressive, et peuvent être effacés de la même manière. Il faut souligner que cette fonctionnalité est intégrée à CSS3, mais ne fonctionne bien sûr que pour les navigateurs compatibles.

Trois méthodes d'animation en volet : slideDown(), slideUp() et slideToggle()

Nous n'entrerons pas dans les détails de tous les paramètres de ces méthodes, pour ne considérer que l'animation produite. Nous utiliserons effectivement les deux premières (**slideDown() et slideUp()**) pour afficher et masquer une liste au survol (ou au clic) des éléments. Cela sera par ailleurs suffisant pour créer de véritables menus déroulants.

Mise en place html-css

page6.html

```
<!DOCTYPE html>
<html>
<head>
     <meta charset='utf-8'>
     <link rel='stylesheet' href='page6.css'>
</head>
<body>
     <ul class='navigation'>
          <li>Section 1</li>
          <ul id='d1'>
               <li>Texte 1</li>
               <li>Texte 2</li>
               <li>Texte 3</li>
          </ul>
          <li>Section 2</li>
          <ul id='d2'>
               <li>Texte 1</li>
               <li>Texte 2</li>
               <li>Texte 3</li>
          </ul>
     </ul>
</body>
</html>
```

Le menu est principalement constitué d'une liste (**ul**) à laquelle est affectée une classe (navigation); cette classe sera utilisée à la fois par la mise en forme css et pour la sélection jquery. Cette liste contient ensuite une alternance de 2 **li/ul**. Les **li** seront les titres des menus, tandis que les **ul** constituerons les sous-menu. Après la mise en place d'une feuille de style basique, nous proposerons une solution permettant de gérer les menus au survol.

La feuille css : page6.css

```
@charset "utf-8";
/* CSS Document */

.navigation{
     width:100px;
}

ul{
     list-style-type:none;
}

#d1{
     background-color:lightgrey;
}

#d2{
     background-color:lightblue;
}
```

D'abord, faire disparaître les listes au chargement ! Notez que l'on masque les **ul** avec jquery, alors qu'on pourrait le faire avec css (display:none), sauf que les utilisateurs inhibant javaScript (il en existe..) ne pourraient pas afficher de nouveau ces listes, et donc les sous-menus qui ne seraient jamais accessibles!!

Mettez-en place le code suivant :

```
<script src='http://ajax.googleapis.com/ajax/libs/jquery/1.10.2/jquery.min.js'>
</script>
<script>
$(document).ready(function(){
     // cacher les listes à l'intérieur des éléments de classe navigation
```

```
});
</script>
```

Notez que si le commentaire explicite la ligne de code, celle-ci n'est pas écrite, il vous appartient donc de le faire !!
Ajoutez la ligne de code correspondante.

Ensuite, ajouter le comportement de survol sur les éléments li. Il s'agit donc dans un premier temps de filtrer les éléments **li**. La difficulté consiste à ne prendre que les **li constituant les titres** :

```
<ul class='navigation'>
     <li>Section 1</li>
     <ul id='d1'>
       <li>Texte 1</li>
       <li>Texte 2</li>
       <li>Texte 3</li>
     </ul>
```

Plusieurs solutions s'offrent à nous : utiliser une classe, un id, ou bien encore la hiérarchie en place. Cette dernière expression consiste à désigner un élément en utilisant sa position relative à d'autres éléments - parent, enfant, suivant, etc.. Ainsi, on peut remarquer que nos éléments **li** de titrage sont **enfants directs** de l'**ul** de classe navigation. Cette sélection s'écrit en css comme en jquery : **.navigation > li**
Le symbole > définit en effet la parenté directe **- Selecteur enfant (“parent > enfant”)**
Ainsi, nous affecterons le comportement **mouseenter** sur ces éléments de la manière suivante :

```
$('.navigation > li').on({
     mouseenter :function(){
          $('.navigation ul').hide();
          $(this).next().slideDown();
     }
});
```

Ce comportement prévoie qu'à l'entrée du curseur dans l'élément (**mousenter**) :

- on cache tous les **ul** à l'intérieur du conteneur de classe navigation ;
- on affiche l'élément qui suit (**next()**) celui sur lequel le curseur est entré.

Ce dernier filtrage permet en effet de cibler l'**ul** qui suit directement le **li** correspondant !!

Modifiez le code selon ce qu'on vient d'expliquer :

```
<script>
$(document).ready(function(){
      $('.navigation ul').hide();
      $('.navigation > li').on({
           mouseenter :function(){
                $('.navigation ul').hide();
                $(this).next().slideDown();
           }
      });
});
</script>
```

Enfin, le masquage des éléments lorsqu'on quitte le menu.

Le menu doit rester affiché tant qu'on est dans le li ou l'**ul** qui le suit. Pour **li**, pas de problème, le comportement **mouseenter** gère cet aspect. Par contre, nous devons prévoir le masquage du sous-menu (**ul**) dès que l'on en sort :

```
      $('.navigation ul').on({
           mouseleave :function(){
                $(this).slideUp();
           }
      });
```

Modifiez le code en conséquence.

Ainsi, dès que l'on quitte une liste à l'intérieur de l'**ul** de classe navigation, on masque cette liste en mode **slideUp**. Notez que ce comportement devrait aussi être affecté à l'**ul** principale, puisque lorsqu'on quitte le menu principal, il est normal que le sous-menu éventuellement ouvert se rétracte entièrement.

Proposez le code correspondant → une piste : les **ul** à masquer sont des enfants de cette **ul** principale!!