**Rapport Projet Javascript-Sélection**

Mon sujet consistait à reproduire un plugin jQuery UI existant (visible au lien suivant :). Ce plugin regroupe plusieurs fonctionnalités permettant de gérer la sélection d’éléments contenues, ainsi :

* Il faut pouvoir sélectionner un élément à l’aide d’un simple clic à la souris, un seul élément est sélectionnable à la fois
* Pouvoir sélectionner plusieurs éléments si la touche ctrl est enfoncée
* Pouvoir récupérer les éléments sélectionnés
* Pouvoir sélectionner plusieurs éléments à l’aide d’une zone de sélection

**Contextualisation**

Le script Javascript réalisé est fait pour fonctionner à l’origine sur une liste à puce (ul) permettant de sélectionner des « li ». Chaque « li » possèdent une classe en fonction de leurs états. Si le « li » n’est sélectionné, il possède la classe « nselect ». S’il est sélectionné ce sera la classe « select ». Le seul changement entre les deux éléments se situera au niveau du CSS. Dans le cas présenté seul la couleur du background change.

**Sélection d’un élément**

La sélection d’un élément est sans doute la partie la plus simple. Pour cela il faut simplement détecter un clic sur l’un des « li » appartenant à un « ul ». Si un clique est détecter sur un « li », on désélectionne, s’il y en a, tous les éléments sélectionnés (fonction del\_select()), et l’on ajoute la classe « select » sur le « li » concerné (fonction select\_deselect()). L’opération sera répétée à chaque nouvelle détection.

**Sélection Multiple**

La sélection multiple implémente une condition supplémentaire lorsque l’on cherche à sélectionner un élément. Il faut vérifier si la touche « ctrl » est pressé lorsque l’on clique sur un « li ». Pour ce faire nous détectons à chaque touche pressée si celle-ci est pressée (ctrl correspond à la valeur 17). Afin d’avoir accès à l’information au sein de l’intégralité du script, nous définissons une variable globale « ctrl », qui est un booléen, à false. Si la touche pressée est ctrl la variable « ctrl » sera initialisé à vrai. Si la touche relâchée est ctrl, la variable est remise à false. Ainsi lors de chaque clic, une condition vérifiera l’état de cette variable. Si elle est à true, uniquement la fonction select\_deselect() sera appelée. Nous avons aussi dû ajouter à cette même fonction une condition vérifiant la classe du « li » où se situe le clic, permettant d’alterner les classes et donc d’autoriser aussi bien la sélection que la désélection lorsque nous sommes en mode sélection multiple.

**Sélection par zone**

Parler de la div en général , CSS ,

La sélection par zone est une fonctionnalité mélangeant deux éléments complexes, la gestion de la zone de sélection et la gestion des collisions. Commençons tout d’abord par la zone de sélection, pour pouvoir créer une zone sélection il faut que le clic de la souris soit maintenu. Pour cela nous avons créé une variable mousedownbool qui est un booléen permettant de contrôler si la souris est pressée. Lorsque la souris est pressé nous initialisons donc une div avec comme coordonnées (top, left) les coordonnées de l’emplacement du click de la souris. Puis nous affichons cette div. Le deuxième contrôle s’effectue lors du déplacement de la souris. En effet en fonction du déplacement de la souris et donc de sa position nous devons adapter la taille de la div. C’est la méthode sizeSelectBox() qui est en charge de cette tâche. Cette méthode va dans un premier temps récupérer la position actuelle de la souris et contrôler si ces positions sont bien supérieur aux coordonnées à l’origine dont les coordonnées sont stockées dans (yPosInverseOrigin & xPosInverseOrigin). Si cette condition est respectée, l’origine de la div à savoir le coin haut-gauche ne sera pas modifié (restera la coordonnée originel ou la souris a été pressée). Par contre si l’origine de la div est supérieur à la position de la souris, là un traitement sera effectué. On alternera les coordonnées de l’origine de la div (top,left) sera modifié par les valeurs des coordonnées de position de la souris à un instant T(x,y). En effet, il est impossible de d’instancier une div avec des valeurs de height ou width negative. Les tests sont bien entendu séparé en deux, un pour chaque point de la coordonnée(x,y) pour faire face à tous les cas possibles. Les variables (xSelectBoxInverse,ySelectBoxInverse) booléennes servent uniquement à empêcher un changement d’origine intempestif. Un changement d’origine ne s’effectuera que si nous ne sommes pas déjà dans le cas d’une alternance de coordonnées (valeur à false). Ainsi les variables (xSelectBoxInverse,ySelectBoxInverse) ne servent uniquement qu’à sauvegarder l’origine de la div afin de la restaurer si la taille de la div redevient positive. Ce sont donc de pures variables de stockage global. Il ne reste qu’à faire le calcul, la différence entre les coordonnées(absolu) et à modifier le css de la div concernée(y->height,x->width). Il ne reste plus qu’à gérer le cas où l’utilisateur relâche le bouton de la souris, en effet la div de sélection est uniquement active lorsque le bouton de la souris est maintenu. Ainsi, il suffira juste de restaurer la taille de la div à 0 est de dissimuler celle-ci (hide()). La gestion des collisions est maintenant le second problème sur lequel je me suis atteler et qui pour moi a été la plus complexe, notamment afin de trouver une formule gérant tous les cas possibles. C’est la fonction collisionSelectBox() qui est en charge des collisions. Elle parcourt chaque élément sélectionnable récupère leurs coordonnés (Bord Haut-Gauche, Haut-Droite,Bas-Gauche,Bas-Droite). Puis les compare avec les coordonnées de la div de sélection à l’aide d’une simple formule de comparaison (formule qui n’est pas de moi, que l’on retrouve dans plusieurs cours traitant de la gestion de collision, assurant l’optimisation et traitant tous les cas possible). Ainsi s’il n’y a pas de collision entre eux la fonction collision\_nselect(e) est appelé qui lui enlève la classe « select » et la remplace classe « nselect ». La fonction ncollision\_nselect(e) agit s’il y a une collision.