Tuto Algo

L’algo de tri fonctionne de la façon suivante :

La fonction alignAll appelle récursivement la fonction align qui va placer chaque génération d’enfants. #larécursivitéc’estlefutur

On attribue à chaque élément un delta correspondant par défaut à leur taille (left ou top selon le classement) ou correspondant par extension à l’espace maximal qu’occupe ses derniers enfants (appelées feuilles).  
Les premiers enfants d’une génération sont placés à l’extrémité voulue du delta de leur père moins leur propre delta. Chacun des enfants suivants de la même génération sont placés suivant le demi-delta de l’enfant précédent additionné à leur propre demi-delta. #larécursivitéc’estlefutur  
  
Exemple : un arbre simple d’un père et de deux fils en classement horizontal classique.  
L’algo calcule l’espace left(ou delta) que les 2 enfants vont prendre.  
Le top des fils prend le top du père plus son épaisseur.  
Le left du 1er fils prend le left du père moins le demi-delta du père plus son propre demi-delta (ici, son delta correspond à son épaisseur puisqu’il n’a pas de fils).  
Le left du 2eme fils prend le left du premier, rajoute le demi-delta du 1er fils et ajoute son propre demi-delta.

///////////////////////////////////

// FONCTIONS

///////////////////////////////////

Les fonctions de base permettre de saisir aisément un élément ou un groupe utiles d’élément.  
  
Les plus utiles sont les suivantes :  
  
//retourne l'objet élément à partir de son id

GetElement : function(idelement)

//retourne l'objet père d'un élément à partir de son id

GetFatherreal : function(idelement, root)

//retourne la 1ere génération d'enfants d'un élément (le TabChild)

GetChilds : function(father)

//retourne toutes les générations de fils d’un élément (le TabAllChild)

GetAllChilds : function(father)

//permet d'avoir un tableau des éléments ascendants d'un élément défini jusqu'au root

GetAscendents : function (oneElement, TabAllChild, root){

//Permet d'obtenir le level d'un élément en comptant le nombre de ses ascendants jusqu'au root

GetLevel : function (oneElement, TabAllChild, root){

// Permet d'obtenir les feuilles de la 1ere génération d'un élément (inutile seule mais est appelée

// récursivement par GetAllFeuilles)

GetFeuilles : function(tableau, TabFeuilles)

//permet d'obtenir toutes les feuilles d'un élément

GetAllFeuilles : function(father)

//retourne la largeur de toutes les feuilles d'un élément selon la largeur de base entre éléments

GetWidthAllFeuilles : function(father, root, TabAllChild, espaceLeftH)

//retourne le résultat de l'addition du width de tous les derniers fils et de l'espace entre eux

GetHeightAllFeuilles : function(father, root, TabAllChild, espaceTopH)

///////////////////////////////////

// TEST

///////////////////////////////////

Prérequis :   
- Un dossier contenant :   
 test.html

TreeClassification.js

Jquery-2.1.1.js  
  
1) Lancer test.html sous un navigateur

2) Lancer TreeClassification.js et test.html sous un éditeur

3) **Pour lancer l’algo avec un arbre aléatoire (random) :**

* Dans TreeClassification.js, à la dernière fonction « alignAll », décommenter les lignes :

var elemen = TreeClassification.TreeRandom(LevelMax,ChildMax) ;

TreeClassification.animate(Final, "blue")

TreeClassification.animateTraitFatherChild(Final,"blue",elemen)

TreeClassification.animateTrait(Final,"red")

Et mettre en commentaire la ligne :

TreeClassification.animate(Final, "")

* Retirer aux paramètres de la fonction « alignAll » :

elemen  (qui sera ici généré par la fonction TreeRandom)

* Ajouter aux paramètres de la fonction « alignAll » ceux necessaires au TreeRandom :

LevelMax, ChildMax

On doit donc obtenir :

alignAll : function(root, espaceLeftH, espaceTopH, boolvertical, boolmirror, booldossier, booldossiermirror, boolcompact, LevelMax, ChildMax)

* Attrapper sauvagement son navigateur (encore resté sur test.html) et entrer sans respect dans la console :  
    
  TreeClassification.alignAll(“A”, 10, 10, false, false, false, false, false, 5, 4).

Félicitations, vous avez généré votre premier arbre random. #leanimatec’estsexy  
  
PS : Les traits nécessitant un canevas de 10000\*10000, il faut donc s’amuser à chercher l’arbre.

PS2 : Pour refaire un deuxième arbre, il faut refresh la page.

PS3 : à partir de paramètres finaux en 5, 5 (soit 5 générations de 5 enfants max), ça rame sec.

4) **Pour lancer l’algo avec une table d’éléments prédéfinis (custom) :**

- Décommenter la création du tableau elemen dans le fichier test.html

- Décommenter dans TreeClassification.js :  
  
TreeClassification.animate(Final, "blue")

TreeClassification.animateTraitFatherChild(Final,"blue",elemen)

TreeClassification.animateTrait(Final,"red")

-Recommenter si necessaire :  
  
var elemen = TreeClassification.TreeRandom(LevelMax,ChildMax)

* Faire ressembler la fonction align à ça:

alignAll : function(elemen, root, espaceLeftH, espaceTopH, boolvertical, boolmirror, booldossier, booldossiermirror, boolcompact)

* Attrapper sauvagement son navigateur (encore resté sur test.html) et entrer sans respect dans la console :  
    
  TreeClassification.alignAll(elemen, “A”, 10, 10, false, false, false, false, false)

Félicitations, vous avez généré votre premier arbre custom.

PS : Plus besoin de s’amuser à chercher l’arbre, il est tout en haut à gauche.  
PS2 : Pour refaire un deuxième arbre, il faut refresh la page.

**5) Pour s’amuser en toute sécurité avec les paramètres :**

* Avec l’arbre random :

Horizontal normal : TreeClassification.alignAll(“A”, 10, 10, false, false, false, false, false, 5, 4)

Horizontal compact : TreeClassification.alignAll(“A”, 10, 10, false, false, false, false, true, 5, 4)

Horizontal miroir : TreeClassification.alignAll(“A”, 10, 10, false, true, false, false, false, 5, 4)

Horizontal dossier : TreeClassification.alignAll(“A”, 10, 10, false, false, true, false, false, 5, 4)

Horizontal dossier2 : TreeClassification.alignAll(“A”, 10, 10, false, false, true, true, false, 5, 4)

Vertical normal : TreeClassification.alignAll(“A”, 10, 10, true, false, false, false, false, 5, 4)

Vertical compact : TreeClassification.alignAll(“A”, 10, 10, true, false, false, false, true, 5, 4)

Vertical miroir : TreeClassification.alignAll(“A”, 10, 10, true, true, false, false, false, 5, 4)

Vertical dossier : TreeClassification.alignAll(“A”, 10, 10, true, false, true, false, false, 5, 4)

Vertical dossier2 : TreeClassification.alignAll(“A”, 10, 10, true, false, true, true, false, 5, 4)

* Avec l’arbre custom :

Horizontal normal : TreeClassification.alignAll(elemen, “A”, 10, 10, false, false, false, false, false)

Horizontal compact : TreeClassification.alignAll(elemen, “A”, 10, 10, false, false, false, false, true)

Horizontal miroir : TreeClassification.alignAll(elemen, “A”, 10, 10, false, true, false, false, false)

Horizontal dossier : TreeClassification.alignAll(elemen, “A”, 10, 10, false, false, true, false, false)

Horizontal dossier2 : TreeClassification.alignAll(elemen, “A”, 10, 10, false, false, true, true, false)

Vertical normal : TreeClassification.alignAll(elemen, “A”, 10, 10, true, false, false, false, false)

Vertical compact : TreeClassification.alignAll(elemen, “A”, 10, 10, true, false, false, false, true)

Vertical miroir : TreeClassification.alignAll(elemen, “A”, 10, 10, true, true, false, false, false)

Vertical dossier : TreeClassification.alignAll(elemen, “A”, 10, 10, true, false, true, false, false)

Vertical dossier2 : TreeClassification.alignAll(elemen, “A”, 10, 10, true, false, true, true, false)

Les possibilités d’arbres sont plus importantes que cela car les fonctions peuvent s’empiler.  
5 booléens à false ou true, 5²=25possibilités. Ça devient hardcore pour faire des screens…

///////////////////////////////////

// PRODUCTION

///////////////////////////////////

Prérequis :

TreeClassification.js

A ajouter a bbmap\_main.txt

* Ajouter le contenu de » A ajouter a bbmap\_main.txt » à l’endroit convenu dans le bbmap originel.
* Remplacer le TreeClassification.js dans le dossier vendors.  
  Par défaut le TreeClassification.js est prêt pour la prod.

A tout hasard, vérifier que les fonctions suivantes de alignAll sont COMMENTEES :  
  
 //var elemen = TreeClassification.TreeRandom(LevelMax,ChildMax)

//TreeClassification.render(TreeClassification.elements, "red");

//TreeClassification.animate(Final, "blue")

//TreeClassification.animateTraitFatherChild(Final,"blue",elemen)

//TreeClassification.animateTrait(Final,"red")

Et celle-ci DECOMMENTEE :

TreeClassification.animate(Final, "")

Les jongles avec les booléens restent les mêmes :

Pour un arbre horizontal enfants en haut :

var vertical = true

Pour un arbre vertical enfants à droite :

var vertical = true

Pour que les enfants soient classés en symétrie par rapport au root:

var mirror = false

Pour un classement en dossier:

var dossier = true

var dossier2 = false

Pour un classement en dossier, version 2 :

var dossier = true

var dossier2 = true

Pour une version compacte de l’arborisation :

var compact = true

…Toujours 5²=25 possibilités…

Toutes inter opèrent entre elles, aucune règle si ce n’est que le dossier2 ne marchera pas si son frère dossier n’est pas sur « true » !   
Le mieux c’est de tester.

GL &HF !