

# Javascript 1: syntaxe de base

jean-luc.falcone@unige.ch

Série 1 - Février 2018

## 1 Rosiers

Pour cet exercice, on implémentera une structure de données représentant des *rosiers* (Rose trees en anglais) ces arbres ont la particularité d'avoir un facteur de branchement non borné. Les données sont stockées dans les noeuds.

Pour simplifier la structure, on se limitera aux indications suivantes:

- Un arbre ne peut pas être vide.
- Un noeud se compose d'une valeur et d'une liste de sous-arbres enfants.
- Une feuille est un juste un noeud dont la liste de sous-arbres enfants est vide.
- N'importe quel type peut être stocké dans les noeuds.

En notation BNF:

```
<RoseTree> ::= <Value> <Children>  
<Children> ::= Empty | <RoseTree> <Children>
```

### 1.1 Implémentation

Créer les fonctions suivantes qui permettent de créer et d'utiliser les arbres binaires définis ci-dessus

```
/* Retourne un arbre contenant une valeur unique */  
function single( x ) { ... }
```

```

/* Retourne un nouvel arbre issu de la fusion des deux arbres.
   Ceux-ci ne sont pas modifiés */
function join( tree1, tree2 ) { ... }

/* Ajoute une valeur à la droite d'un arbre, celui-ci n'est pas modifié */
function append( tree, x ) { ... }

/* Ajoute une valeur à la gauche d'un arbre, celui-ci n'est pas modifié */
function prepend( tree, x ) { ... }

/* Retourne la taille d'un arbre (nombre de valeurs dans l'arbre) */
function size( tree ) { ... }

/* Retourne la longueur de la branche la plus profonde d'un arbre */
function depth( tree ) { ... }

/* Retourne un tableau comprenant toutes les valeurs de l'arbre
   ordonnées selon un parcours en profondeur */
function toArray( tree ) { ... }

```

Votre implémentation devra respecter les règles suivantes:

```

/* Pour tout arbre A1 et A2, et pour toute valeur X */
size( A1 ) + size( A2 ) === size( join( A1, A2 ) );
max( depth( A1 ), depth( A2 ) ) === depth( join( A1, A2 ) );
size( append( A1, X ) ) === size( A1 ) + 1;
size( prepend( A1, X ) ) === size( A1 ) + 1;
toArray( append( A1, X ) ) === toArray( A1 ).concat( [X] );
toArray( prepend( A1, X ) ) === [X].concat( toArray( A1 ) );
size( A1 ) === toArray( A1 ).length;
toArray( A1 ).concat( toArray( A2 ) ) === toArray( join( A1, A2 ) );

```