Les bases du langage JavaScript - Quelques éléments
Source : JavaScript - The definitive guide - O'Reilly
Auteur : G. SIMON - Département Informatique - IUT du Havre

1. Syntaxe de base

- commentaires : /* ou //
- 3 types de données primitifs : nombres, chaines et booléens
- tous les nombres sont codés comme des réels
- on peut coder des nombres en chaines (methode toString()) et vice-versa (methode parseInt(ch) ou parseFloat(ch))
- chaines de car : "toto" ou 'toto'
- les caractères sont représentés par des chaines de longueur 1
- concaténation de chaines : opérateur +
- longueur d'une chaine : attribut length
- méthodes sur les chaines : charAt, substring, indexOf
- booléens : true et false
- valeurs spéciales : null (une variable qui n'a pas de valeur) et undefined (une variable qui n'existe pas)
- objets parmi lesquels fonctions et tableaux

2. Les objets

- création
    - var point = new Object(); point.x = 2; point.x = -1;
    - var point = { x:2, y:-1 };
    - dans les 2 cas, l'objet a une structure qui est définie à la volée
- on peut détruire une propriété d'un objet (delete obj.prop)
- fonction instanceof : verifie le nom du constructeur utilisé pour créer un objet
- fonction typeof : permet de connaitre le type d'une variable (objet,booleen,chaine,nombre,fonction). NB : le type d'un tableau est "object".
- methode valueOf() présente dans certains objets permet de "convertir" l'objet en une valeur d'un autre type.

3. Les tableaux

- tableaux simples vs tableaux associatifs (ces derniers sont en fait des objets : les clefs sont les noms des variables d'instance et les valeurs sont les valeurs des variables d'instance). Les tableaux simples sont aussi représentés par des objets.

- attention : les indices des tableaux simples sont forcément des entiers positifs. Donc si on écrit par exemple t[-1.23] = true alors ça va définir une nouvelle propriété appelée "-1.23" à l'objet représentant le tableau.

- NB : les indices d'un tableau simple ne sont pas forcément contigus. On peut avoir un tableau avec 2 cases allouées dont les indices sont 0 et 999. Mais dans ce cas la longueur (attribut length) du tableau est 1000.

4. Valeur vs Référence
- entiers et booléens sont tout le temps manipulés par valeur
- les objets (et donc les tableaux et les fonctions) sont manipulés par référence
- les chaines sont considérées "immutable" pour la copie et le passage de paramètre. Elles sont en revanche comparées par valeur (avec ==) ce qui est différent de JAVA.

5. Opérateurs d'égalité et de différence
- opérateurs d'égalité
    - === (identité) : équivalent au == de Java
    - == (égalité) : commme === + fonctionne aussi pour des types différents moyennant des conversions

- opérateur de différence : !== (négation de ===) et != (négation de ==)

NB : il faut préférer === à ==

6. Structures de controle
- if
- switch
- while
- do while
- for var in Object (pour parcourir les propriétés d'un objet)

7. Gestion des variables
- les variables déclarées en dehors de toute fonction sont ajoutées comme propriétés à l'objet principal implicite (accessible par this). Dans des appli web, l'objet principal s'appelle window.

- une var déclarée dans une fonction est ajoutée comme propriété à l'objet de type call créé automatiquement à chaque exécution de fonction.

8. Les fonctions
- on peut définir une fonction nommée ou non nommée

- attention : appeler une fonction avec un mauvais nombre de parametres ne declenche pas une erreur ! (voir propriete arguments)
    → lorsqu'il y a moins de paramètres effectifs que de params formels, les params formels non associés ont la valeur undefined
    → lorsqu'il y a plus de paramètres effectifs que de params formels, les params effectifs supplémentaires sont accessibles (ainsi que tous les autres) via le tableau "arguments".

- lorsqu'une fonction est associée à une propriété d'un objet, on peut utiliser this dans la fonction

- les fonctions sont des objets à part entière et ont donc des propriétés :
    - length : nb de paramètres attendus
    - prototype : cet attribut a comme valeur un objet représentant le prototype associé lorsque la fonction est un constructeur. Cet objet a une seule propriete appelee constructor dont la valeur est la fonction elle-même (l'objet representant cette fonction)