Plan de la leçon

1. Infos (http://www.timcsf.ca/styles-codage/); Présentation du plan de cours;
2. Révision de la configuration de l'éditeur pour TypeScript;
3. Contexte de développement et patterns de performance;
4. ES2015 – Le noyau - La portée des variables;
5. Laboratoire : exercice formatif, les méthodes des tableaux [Récupérer de Léa et déposer sur Léa avant le prochain cours];

3. Contexte de développement

3.1 Web… le DOM

Référence principale pour explorer le DOM : https://developer.mozilla.org/fr/docs/Web/API

diagramme 1 – level 0

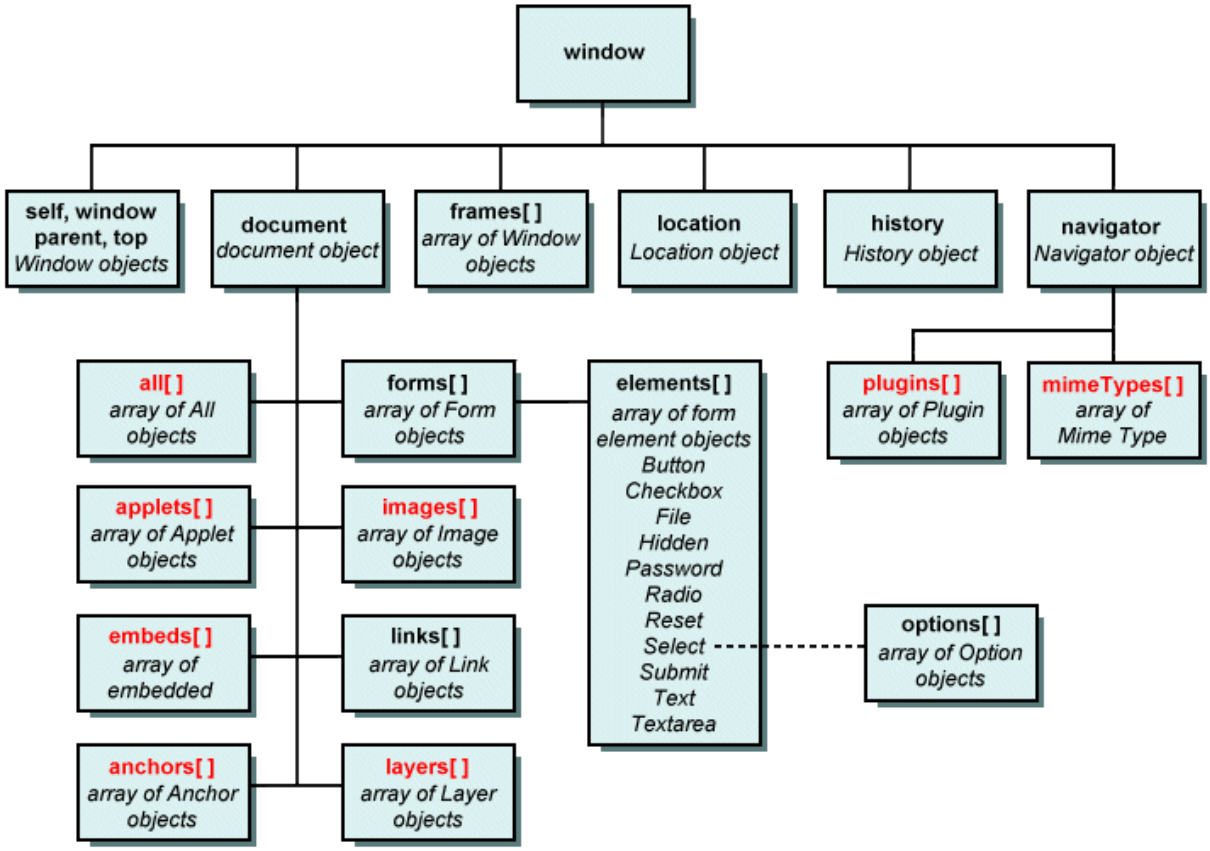
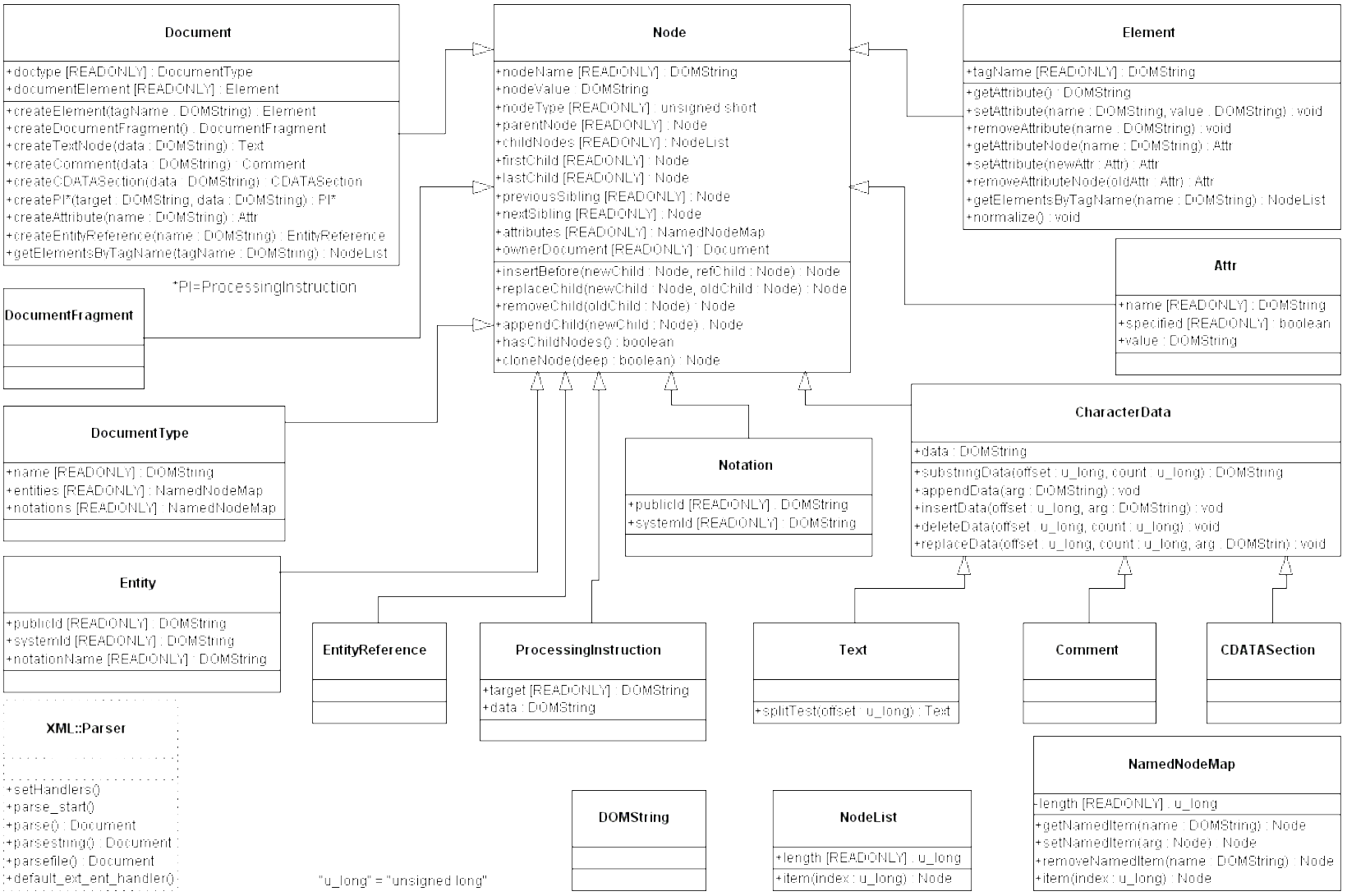


diagramme 2 – level 1 (page suivante)

Logical View

DOM (Core) Level One

Invoke "get*Name*" to read instance variable *name* when using XML::DOM or XML4J



3.2 Patterns de performance

Bien que les liens entre le DOM et le JavaScript soient très étroits, on pourrait même dire « intimes », certains aspects de cette relation peuvent être « lourds » surtout en raison des ramifications de l'arborescence HTML qui peuvent devenir touffues. Ainsi, un simple document.querySelector(".uneClasseX") peut s'exécuter très rapidement dans une petite page alors que dans un document de quelques milliers de lignes avec des hiérarchies profondes, cette même ligne de code prendra pas mal plus de temps. Bien sûr ce « pas mal plus de temps » reste de l'ordre d'une fraction de seconde, toutefois, plus les tâches à réaliser en JavaScript se complexifie, plus ces mini-pertes de temps risque de s'accumuler et de provoquer de véritables périodes de latence ayant un impact sur la qualité de l'expérience de l'utilisateur.

En conséquence il est prudent de se pencher tout de suite sur quelques pratiques (inspirés de ***JavaScript Patterns*** de Stoyan Stefanov) qui pourront minimiser ces risques. En voici un aperçu que nous approfondirons plus tard :

- **Limiter le nombre de globales**
    - [DÉSUET] Utiliser un espace de nom
      (IIFE : http://benalman.com/news/2010/11/immediately-invoked-function-expression/)
    - [ACTUEL] ES2015 : **Développement modulaire**
      et privilégier l'utilisation d'un bloc pour ne pas encombrer le Window
- **Limiter les accès au DOM**
    - stocker dans des variables les références des objets du DOM
      Exemple :
      ```
      private refMenuListe: Element = document.querySelector(".menu__liste");
      ```
    - éviter l'accès au DOM ds les boucle
    - etc.
- **Limiter le nombre de modifications au DOM qui oblige le rafraîchissement du rendu par le navigateur**
    - au lieu de modifier au fur et à la mesure, y aller en *batch* avec `createDocumentFragment()`

4. ES2015 – le noyau

4.1 La portée des variables

Voir démo.

var, let, const et freeze.

var

- on peut changer la valeur d'une variable déclarée avec var et on peut aussi la re-déclarer sans problème
- portée limitée à l'intérieur d'une fonction mais si définie en dehors d'une fonction, la portée est celle d'une variable globale.

let et const

- portée limitée à un bloc. Qu'est-ce qu'un bloc? Tout ce qui se trouve entre des accolades {}.

let

- on peut changer la valeur d'une variable déclarée avec let mais on **ne** peut **pas** la re-déclarer

const

- on **ne** peut **pas** changer la valeur d'une variable déclarée avec const
  Par contre on peut en modifier les propriétés s'il s'agit d'un objet! Si on veut rendre impossible les modifications de propriétés, on peut utiliser Object.freeze().
- on **ne** peut **pas** la re-déclarer

4.2 Les méthodes des tableaux

Voir démo.

splice(), sort(), push(), pop(), shift(), unshift(), join(), reverse(), slice(), concat(), filter(), map(), forEach(), reduce()

Référence : https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets_globaux/Array

Rappel concernant **l'instruction ternaire ou opérateur conditionnel :**
https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Opérateurs/L_opérateur_conditionnel