# 1. DOM ET DHTML

## 1.1. Présentation

**DHTML** est avant tout un **terme médiatique** qui indique simplement que certains éléments de la page peuvent être dynamiquement modifiés par des morceaux de scripts (JavaScript, ...).

Anciennes choses obsolètes :
**document.~~layerA~~.document.** avec **Netscape 4** .
**document.~~all~~.** avec **IE4**.

Standard à suivre :
Le **W3C (World Wide Web Consortium)** a petit à petit élaboré un **modèle objet normalisé** (DOM de Niveaux 1,2 et 3) avec entre autres "**document.getElementById()"** . Ce modèle objet normalisé s'appuie sur l'arborescence générique d'XML et sur les feuilles de style CSS.

# 2. Le modèle normalisé (DOM du W3C) (depuis IE5)

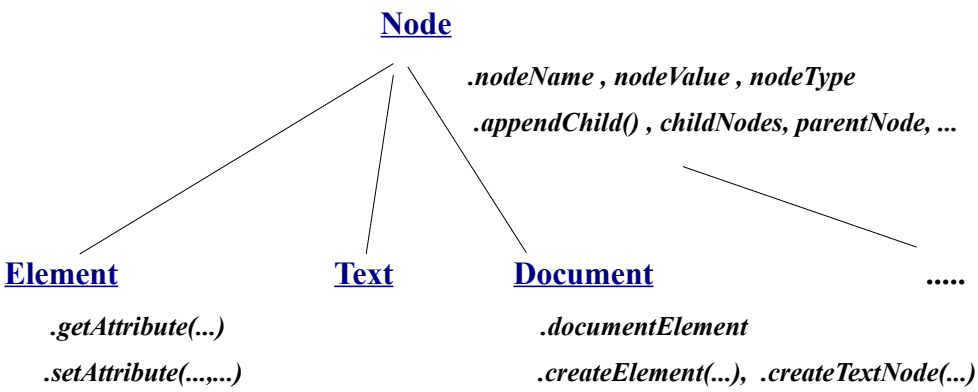
Le modèle objet normalisé par le W3C est complètement documenté au bout de l'URL suivante: ***http://www.w3.org/DOM/***
HTML peut être vu comme un cas particulier de XML . On parle de **XHTML**.
Le **DOM niveau 1 (Core)** s'applique à **XML**
Le **DOM niveau 1 (Html)** s'applique à (X)**HTML**.
Le **DOM niveau 2** s'applique essentiellement aux **styles (CSS)** et aux **événements**.

## 2.1. DOM Niveau 1 (Core)

**XML DOM** est une **API normalisée** au niveau du W3C.
L' api XML DOM permet de **construire un arbre en mémoire**. Chaque noeud de l'arbre correspond à un élément XML quelconque (Balise, Zone Textuelle, Instruction de traitement, ...).
Dans un arbre DOM, les noeuds ne sont pas tous du même type:

- Le noeud racine est de type **Document**
- Les noeuds liés aux balises sont de type **Element**
- Les noeuds comportant un morceau de texte sont de type **Text**

Cependant , ces différents types de noeuds héritent tous d'un type générique : "**Node**" .



Certaines propriétés (associées au type générique ***Node***) sont accessibles depuis n'importe quel noeud :

- ***nodeName*** retourne le nom d'une balise ou null .
- ***nodeValue*** retourne la valeur d'un texte ou null .
- ***nodeType*** retourne une constante indiquant le type de noeud (ELEMENT_NODE , TEXT_NODE, ....)
- **childNodes** retourne une liste de noeuds fils sous la forme d'un objet ensembliste de type **NodeList** .
- **parentNode** retourne une référence sur le noeud père

D'autres fonctions ne sont disponibles que sur certains types précis de noeuds:

- La fonction **documentElement()** que l'on appelle sur le noeud racine du document retourne **l'unique noeud de type Element qui correspond à la balise de premier niveau .**
- Seul le noeud racine (de type *Document*) comporte des fonctions [ ***createElement***("nombalise") , ***createTextNode***("valeurTexte") ] permettant de créer de nouveaux noeuds qui devront ultérieurement être accrochés sous un des noeuds de l'arbre via la méthode ***appendChild***() .
- Les méthodes ***setAttribute***("nomAttribut","valeur") et ***getAttribute***("nomAttribut") doivent être appelées sur un noeud de type *Element* .

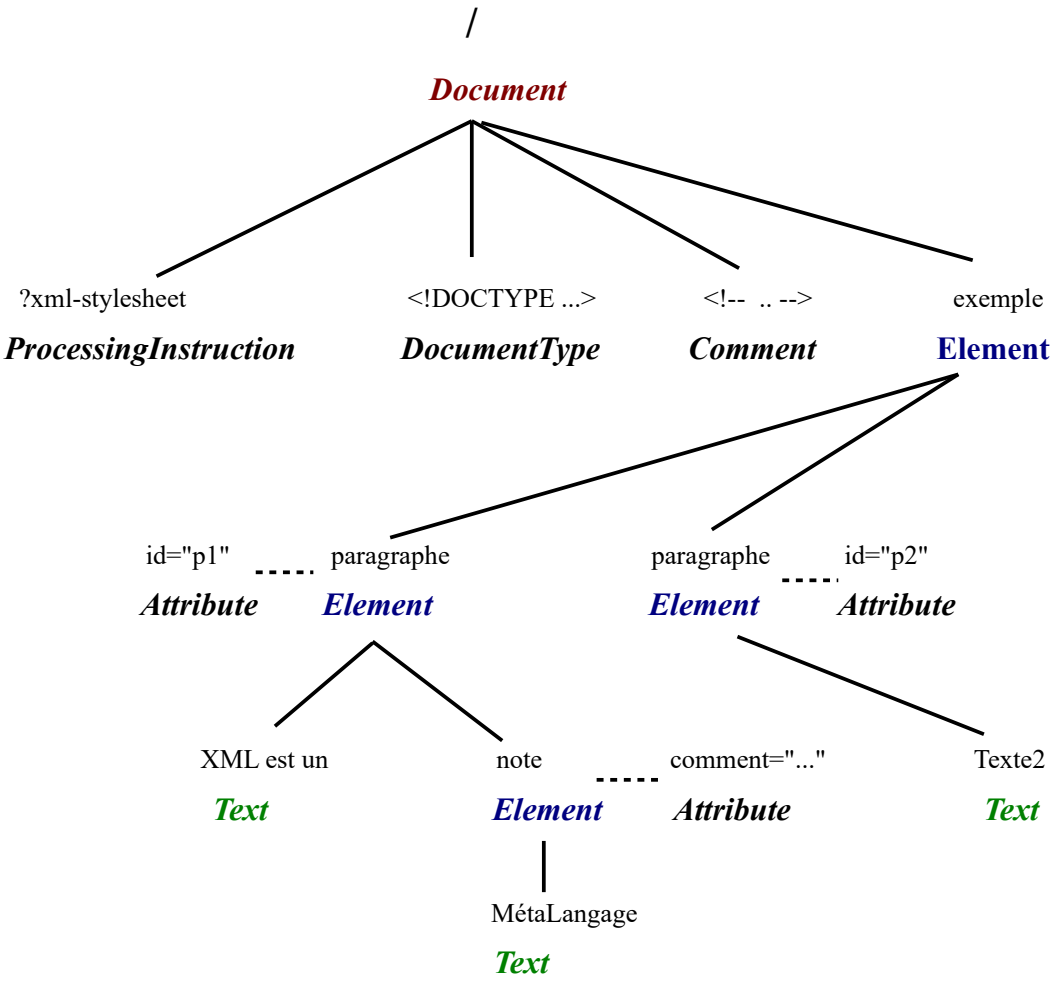
L'interface **Attr(ibute)** correspond à un attribut. *Les noeuds de type Attr(ibute) ne sont pas directement placés sous un noeud de type Element*.
Les différents attributs d'un noeuds sont accessibles depuis la collection attributes d'un élément.
L'interface **NodeList** représente un ensemble ordonné de noeuds.
Sa propriété **length** retourne le nombre d'éléments (pour boucle for de 0 à n-1).
Sa méthode **item(i)** retourne le i éme noeud de la liste.

**Correspondance entre fichier XML et arbre "DOM" en mémoire :**

***Le fichier xml suivant***

```
<?xml version="1.0" ?>
<?xml-stylesheet href="./style1.css" type="text/css" ?>
<!DOCTYPE  exemple
    [
     <!ENTITY  eacute  "&#x00E9;" >
    ]>
<!- - commentaire -->
<exemple>
   <paragraphe  id="p1">XML est un
                        <note comment="important"> M&eacute;taLangage</note>
   </paragraphe>
   <paragraphe  id="p2">texte2</paragraphe>
</exemple>
```

***est représenté via un arbre DOM de ce type:***




## 2.2. DOM Niveau 1 (HTML)

Le DOM niveau 1 (HTML) est une extension du DOM (Core) prévue pour HTML.
Une liste d'éléments sera souvent gérée au travers d'une collection :

*interface* **HTMLCollection** {
readonly attribute unsigned long **length**;
Node **item**(in unsigned long *index*);
Node **namedItem**(in DOMString *name*);
};

Un **HTMLDocument** hérite des spécificités d'un **Document** et possède en plus les propriétés et méthodes suivantes:

*interface* **HTMLDocument** : *Document* {
attribute DOMString **title**;
readonly attribute DOMString **referrer**;
readonly attribute DOMString **domain**;
readonly attribute DOMString **URL**;
attribute *HTMLElement* **body**;
readonly attribute HTMLCollection **images**;
readonly attribute HTMLCollection **applets**;
readonly attribute HTMLCollection **links**;
readonly attribute HTMLCollection **forms**;

readonly attribute HTMLCollection **anchors**;
attribute DOMString **cookie**;
void **open**();
void **close**();
void **write**(in DOMString text);
void **writeln**(in DOMString text);
*Element* **getElementById**(in DOMString elementId);
*NodeList* **getElementsByName**(in DOMString elementName);
};

La principale fonction est **getElementById**() . Celle-ci renvoie une référence sur un élément dont on connaît l'id .
Cette fonction est supportée depuis **IE5** et **Netscape 6** .

Un élément quelconque d'une page HTML sera de type **HTMLElement** (héritant de Element héritant de Node).

*interface* **HTMLElement** : *Element* {
attribute DOMString **id**;
attribute DOMString **title**;
attribute DOMString lang;
attribute DOMString dir;
attribute DOMString **className**;
};

Les différentes interfaces suivantes correspondent à des éléments particuliers d'une page HTML:
**HTMLHtmlElement** racine de la page <html>
**HTMLHeadElement** partie <head>
**HTMLLinkElement** <link>
**HTMLTitleElement** <title>
**HTMLMetaElement** <meta>
**HTMLStyleElement** <style>

interface **HTMLBodyElement** : HTMLElement {
attribute DOMString aLink;
attribute DOMString **background**;
attribute DOMString **bgColor**;
attribute DOMString link;
attribute DOMString **text**;
attribute DOMString vLink;
};

interface **HTMLFormElement** : HTMLElement {
readonly attribute HTMLCollection **elements**;
readonly attribute long **length**;
attribute DOMString name;
attribute DOMString acceptCharset;
attribute DOMString **action**;
attribute DOMString **enctype**;
attribute DOMString **method**;
attribute DOMString target;
void **submit**();
void **reset**();
};
**HTMLSelectElement**
**HTMLOptionElement**
**HTMLInputElement**
**HTMLTextAreaElement**
**HTMLButtonElement**
**…**

Un tableau HTML est vu comme un élément du type suivant:
interface **HTMLTableElement** : HTMLElement {
...

readonly attribute HTMLCollection **rows**;
...
attribute DOMString bgColor;
attribute DOMString **border**;
...
HTMLElement **insertRow**(in long index);
void **deleteRow**(in long index);
};

Ligne d'un tableau:
interface **HTMLTableRowElement** : HTMLElement {
attribute long rowIndex;
...
attribute HTMLCollection **cells**;
...
HTMLElement **insertCell**(in long index);
void **deleteCell**(in long index);
};

Cellule d'un tableau: **HTMLTableCellElement**

## 2.3. Exemples (depuis IE5 et NS6)

document.**getElementById**("xxx").**style.visibility** = "hidden"
document.**getElementById**("xxx").**style.color** = "green"
document.**getElementById**("xxx").**innerHtml** = " yyy" *// non normalisé.*
document.**getElementById**("xxx").**onmouseover**= "...."

*Equivalent de l'ancien document.all de IE4:*
var docContents= document.**getElementsByTagName("*");**

Ajouter un élément dans la page:
var txtNode = document.**createTextNode**("blabla");
var link = document.**createElement**('a');
link.**setAttribute**('href','mypage.html');
link.**appendChild**(txt);
document.getElementById("xxx").appendChild(link);

Remplacement du contenu d'un élément (avec NS6):
*// équivalent de **document.getElementById(eltId).innerHtml=content***
function ***dynamicContentNS6***(elementId,content)
{ if(document.getElementById)
{
rng=document.createRange();
e1=document.getElementById(eltId);
rng.setStartBefore(e1);
htmlFrag = rng.createContextualFragment(content);
**while(e1.hasChildNodes())**
**e1.removeChild(e1.lastChild());**
**e1.appendChild(htmlFrag);**
}
}

## 2.4. Traitement des événements (Normalisé par DOM Niveau 2):

==> ne fonctionne pas toujours avec d'anciennes versions de IE .
***http://www.w3.org/TR/DOM-Level-2-Events/events.html***

Set e= document.getElementById("xxx");
e.**addEventListener**("mouseover",fctShowMsg,false);
*// le booléen "capture" à true signifie à peu près "empêcher d'atteindre la cible prévue*

*// (annuler action par défaut).*
e.**removeEventListener**(-,-,-);
*La fonction de traitement est du type suivant:*

**function** showHomeMsg(**evt**) *// evt est de type W3C Event Object (DOM 2)*
{ //....
}

```
// Introduced in DOM Level 2:
interface Event {
// PhaseType
const unsigned short CAPTURING_PHASE = 1;
const unsigned short AT_TARGET = 2;
const unsigned short BUBBLING_PHASE = 3;
readonly attribute DOMString type;
readonly attribute EventTarget target;
readonly attribute EventTarget currentTarget;
readonly attribute unsigned short eventPhase;
readonly attribute boolean bubbles;
readonly attribute boolean cancelable;
readonly attribute DOMTimeStamp timeStamp;
void stopPropagation();
void preventDefault();
void initEvent(in DOMString eventTypeArg,
in boolean canBubbleArg,
in boolean cancelableArg);
};
```

# 3. Pour les navigateurs assez récents (pas trop anciens)

## 3.1. querySelector

De façon à utiliser une syntaxe proche de jQuery et des sélecteurs css , on peut utiliser querySelector() à la place de getElementById() :

```
//var myP=document.getElementById('myP');
  var myP=document.querySelector('#myP');
```

## 3.2. insertRow() , insertCell()

var newRow = tableElt.**insertRow**(-1 ou …) ;
var newCell = newRow.**insertCell**(0 ou …) ;
newCell.**innerHTML**="Paris ou autre" ;

C'est un équivalent plus rapide de document.***createElement***("tr") ; + ***appendChild***(...)