**DOM : interface de programmation pour les documents XML et HTML. (API)**

**Def :**

**API :** est un ensemble d'outils qui permettent de faire communiquer entre eux plusieurs programmes ou, dans le cas présent, différents langages.

Concept :

grâce au DOM, nous allons pouvoir modifier des éléments HTML (afficher ou masquer un <div> par exemple), en ajouter, en déplacer ou même en supprimer.

**Window :** C’est un objet global qui représente la fenêtre du navigateur.

**Document :** C’est un sous objet de Window, il représente la page web et plus précisément la balise <html>.



Représentation d’une page web sous forme d’arbre

L’objet document possède trois méthodes pour accéder aux éléments html (AKA les balises).

***Attention : Quand on parle des balises HTML en javascript, celles-ci sont des objets, on emploiera donc maintenant le terme d’élément pour les nommer.***

**getElementById() :**

Cette méthode à pour paramètre un attribut ‘id’ qui renvoie dans quel type de balise est cette id.

<div id="myDiv">

<p>Un peu de texte <a>et un lien</a></p>

</div>

<script>

var div = document.getElementById('myDiv');

</script>

Ceci renverra que ‘myDiv’ est dans une balise div.

**getElementsByTagName() :**

Cette méthode permet de renvoyer sous la forme d’un tableau tous les éléments de la même famille.

Ex :

var divs = document.getElementsByTagName('div');

for (var i = 0, c = divs.length ; i < c ; i++) {

alert('Element n° ' + (i + 1) + ' : ' + divs[i]);

}

Cela retourne une collection d’éléments, utilisable comme un tableau.

**getElementsByName() :**

Cetteméthode permet de récupérer seulement les éléments qui possèdent un attribut de type ‘name’.

A n’utiliser que dans les formulaires comme l’attribut name n’est que dans les balises utilisés dans les formulaires.

**Aujourd’hui, de nouvelles méthodes ont été créées pour faciliter la sélection des éléments :**

**Ces deux méthodes prennent pour paramètre une chaîne de caractère : celle-ci doit être un sélecteur CSS, que l’on retrouve dans les feuilles de style (Ex : #menu .item span)**

**querySelector() :**

**renvoie le premier élément trouvé correspondant au sélecteur CSS**

**querySelectorAll() :**

**va renvoyer tous les éléments (sous forme de tableau) correspondant au sélecteur CSS fourni.**

**innerHTML :**

permet d'accéder au contenu d'un élément HTML

Ex : var query = document.querySelector('#menu .item span'),

queryAll = document.querySelectorAll('#menu .item span');

alert(query.innerHTML);

alert(queryAll.length);

alert(queryAll[0].innerHTML + ' - ' + queryAll[1].innerHTML);

**Notions d’Héritage :**

Les sous objets ont des propriétés et des méthodes, ceux-ci héritent des propriétés et des méthodes de leurs parents (Voir diagramme) :



**HTMLDivElement hérite de ses parents**

**Editer des éléments HTML :**

Pour interagir avec les attributs, l’objet Element fournit deux méthodes, getAttribute() et setAttribute(). Le premier récupère l’attribut et le deuxième l’édite.

getAttribute(‘href’);

setAttribute(‘href’, nouveau contenu);

Il faut d’abord enregistrer dans une variable l’element que l’on cherche grâce à getElementById(), et ensuite l’utiliser avec getAttribute.

Ex : var link = document.getElementById(‘myId’);

var href = link.getAttribute(‘href’);

Ajouter ou éditer du HTML :

document.getElementById(‘myDiv’).innerHTML = ‘Nouveau texte’;

// mettre += pour mettre à la suite et ne pas effacer le contenu d’avant.

**innerText et textContent :**

Récupère du contenu sans balisage.