DOM et accès en Javascript

1 Introduction au DOM

DOM est le Document Object Model, ou modèle objet de document. C'est une API pour les documents HTML (et XML). Il fournit une représentation structurelle du document, permettant de modifier son contenu et sa présentation visuelle.

La première implémentation du DOM date de 1998. Depuis deux autres versions ont vu le jour, la dernière datant de 2004.

Toutes les propriétés, méthodes et évènements utilisables par le développeur Web pour manipuler et créer des pages y sont organisés au sein d'objets. (c'est-à-dire l'objet document qui représente le document lui-même, l'objet table qui représente un élément de tableau HTML, et ainsi de suite). Ces objets sont accessibles via des langages de scripts dans la plupart des navigateurs récents, en général Javascript (ECMAscript).

Le World Wide Web Consortium établit un standard pour le DOM, appelé W3C DOM. Il doit permettre, maintenant que les navigateurs les plus importants l'implémentent correctement, de réaliser de puissantes applications multi-navigateurs.

2 Organisation du DOM

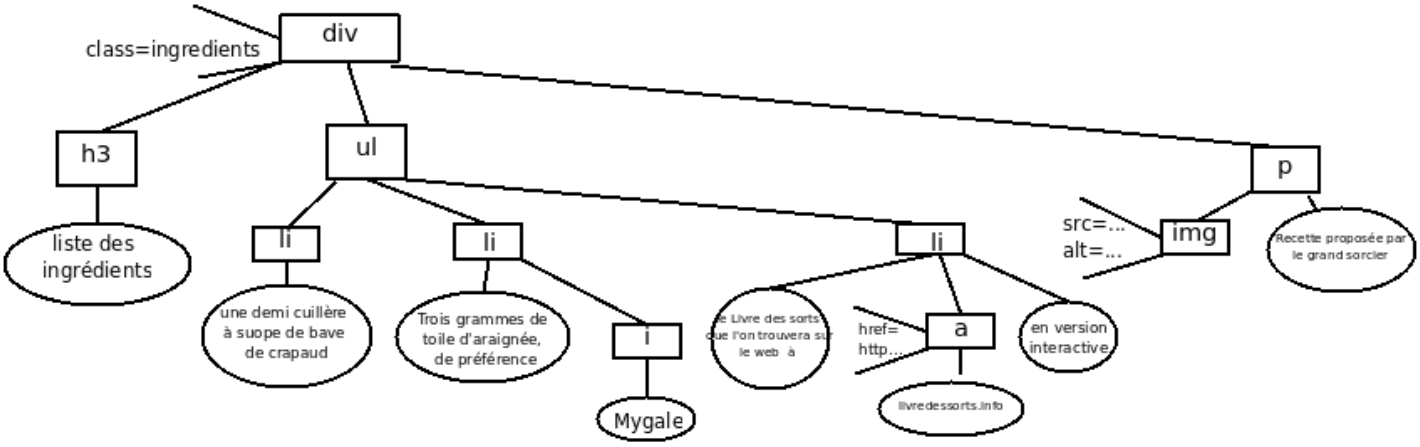
Le programmeur dispose d'objets, qui ont des propriétés, des méthodes et des évènements qui interfacent le document XML ou HTML.

Exemple :

Soit un extrait de code HTML

```
<div class="ingredients">
<h3>Liste des ingrédients</h3>
<ul>
  <li>Une demi-cuillère à soupe de bave de crapaud</li>
  <li>Trois grammes de toile d'araignée, de préférence
<i>mygale</i></li>
  <li>Le livre des sorts (que l'on trouvera sur le web à
     <a href="livredessorts.info">livredessorts.info</a>
     dans une version interactive</li>
</ul>
<p><img src="portraitSorcier.jpg" alt="Portrait du sorcier" />
   Recette proposée par le grand sorcier</p>
</div>
```

Le DOM peut être représenté par l'arbre suivant



Chaque balise ainsi que chaque texte est un nœud de l'arbre et est représenté par un objet dans le DOM.
Chaque propriété de balise est alors un attribut de l'objet.

L'objet de plus haut niveau est l'objet **document**, qui contient l'ensemble des balises HTML du document.

> Remarque : d'où la méthode **document**.write() mainte fois rencontrée en TP pour écrire dans la page en Javascript !

3 Accéder au DOM avec Javascript

Le DOM est le plus souvent utilisé en conjonction avec JavaScript. C'est-à-dire que le code est écrit en JavaScript, mais qu'il utilise le DOM pour accéder à la page Web et ses éléments.

Grâce au DOM, il est possible en Javascript de modifier les CSS d'une balise, d'obtenir la valeur contenue dans un champ de saisie, ajouter des balises au document,...
Pour cela, Javascript fournit un ensemble de méthodes (fonctions en POO).

Les paragraphes suivants illustrent les possibilités.

4 Obtenir la référence à un objet du DOM

Afin de pouvoir modifier un objet du document, encore faut-il savoir comment il se nomme.

4.1 Grâce à l'id

La liaison entre HTML et Javascript se fait par l'utilisation de l'attribut HTML **id**.

Exemple :
```
<input type="text" id="champ1" name="nom" />
```

En Javascript, on récupère la référence à l'élément en appliquant la méthode **getElementById()** à l'objet **document**.

Exemple :
```
var objNom = document.getElementById("champ1") ;
```

> Remarque : il est fréquent en Javascript de créer une fonction $( ) qui permet d'éviter tout ce code. Cette fonction se retrouve dans plusieurs frameworks Javascript (Prototype, Jquery).

Fonction $() :
```
function $(id) {
        return document.getElementById(id) ;
}
```

Le code de récupération d'un élément du DOM est alors plus concis :
```
var objNom = $("champ1") ;
```

4.2 En fonction du nom de balise

De la même façon, on peut récupérer la liste d'objets du DOM qui sont d'un type donné, grâce à la méthode **getElementsByTagName( )** de l'objet **document**. La méthode renvoie les références aux objets dans un tableau.

Exemple : récupération de la liste des balises de type <a>
```
var objNoms = array() ;    //définition d'un tableau des références des objets
objNoms = document.getElementsByTagName("a") ;
```

5 Lecture de propriétés d'une balise

5.1 Cas général

Une fois que l'on a la référence du DOM à l'élément cherché, on peut lire une propriété de la balise. C'est devenu un attribut de l'objet décrivant la balise.

On accède à un attribut par la notation **objet.attribut**

Exemple : récupération de la valeur du href dans une balise <a>
```
var nom=objNom.href ;
```

5.2 Cas des propriétés CSS

On accède aux propriétés CSS en passant par l'objet contenu lui-même dans l'attribut **style** de l'objet :

objet.**style**.proprieteCss

Exemple : récupération de la valeur de la couleur du texte
```
var couleur_nom=objNom.style.color ;
```

> Attention ! De nombreuse propriétés CSS utilisent des tirets (-). Ils sont supprimés dans la notation DOM et remplacés par une majuscule. Par exemple, la propriété CSS *text-align* deviendra *textAlign*

6 Écriture de propriétés d'une balise

On affecte une valeur à la propriété concernée en passant par l'objet lui correspondant dans le DOM :

objet.attribut = 'Une valeur' ;

Exemple1 : changement du nom de fichier d'une propriété *href*
```
objNom.href = "autre.html" ;
```

Exemple2 : changement de la couleur du texte
```
objNom.style.color = "#543" ;
```

7 Ajout de contenu dans la page

7.1 Méthode simple utlisant innerHTML

Il faut utiliser la propriété innerHTML sur l'objet du DOM.

objet.innerHTML = 'code_HTML_à_ajouter' ;

Exemple : ajout d'un paragraphe à une div
Code HTML :
```
<div id="ma_div"></div>
```

Code Javascript :
```
var div=document.getElementById('ma_div') ;
div.innerHTML='<p>Un nouveau paragraphe</p>' ;
```

La page est mise à jour automatiquement avec le nouveau contenu de la .

> En fait, la propriété innerHTML **remplace** le contenu actuel de la balise ciblée par un nouveau. Si un contenu était présent, il est supprimé.

7.2 Ajout de nœud

Utiliser la méthode **createElement("nom_balise")** pour créer une balise ou **createTextNode("texte")** pour créer un texte.
Ensuite utiliser la méthode **appendChild()** pour ajouter l'élément dans le DOM.

Exemple :

```
var c=document.getElementById("conteneur") ;
var titre2=document.createElement("h2") ;       //création d'une balise
h2
c.appendChild(titre2) ;    //ajout de l'objet représentant h2 dans le
conteneur

var t=document.createTextNode("Le titre 2") ;   //création d'un texte
titre2.appendChild(t) ;    //ajout du texte à la balise h2 précédente
```

8 Événements capturables du DOM

8.1 Liste des événements

- événements page et fenêtre
  - *onabort* — s'il y a une interruption dans le chargement
  - *onerror* — en cas d'erreur pendant le chargement de la page
  - *onload* — après la fin du chargement de la page
  - *onbeforeunload* — se produit juste avant de décharger la page en cours (par changement de page, en quittant)
  - *onunload* — se produit lors du déchargement de la page (par changement de page, en quittant)
  - *onresize* — quand la fenêtre est redimensionnée
- événements souris
  - *onclick* — sur un simple clic
  - *ondblclick* — sur un double clic
  - *onmousedown* — lorsque le bouton de la souris est enfoncé, sans forcément le relâcher
  - *onmousemove* — lorsque la souris est déplacée
  - *onmouseout* — lorsque la souris sort de l'élément
  - *onmouseover* — lorsque la souris est sur l'élément
  - *onmouseup* — lorsque le bouton de la souris est relâché
- événements clavier
  - *onkeydown* — lorsqu'une touche est enfoncée
  - *onkeypress* — lorsqu'une touche est pressée et relâchée
  - *onkeyup* — lorsqu'une touche est relâchée
- événements formulaire
  - *onblur* — à la perte du focus
  - *onchange* — à la perte du focus si la valeur a changé
  - *onfocus* — lorsque l'élément prend le focus (ou devient actif)
  - *onreset* — lors de la remise à zéro du formulaire (*via* un bouton "reset" ou une fonction reset())

- *onselect* — quand du texte est sélectionné
- *onsubmit* — quand le formulaire est validé (*via* un bouton bouton de type "submit" ou une fonction submit())

> Remarque : en général, on évite d'utiliser l'événement onclick qui ne provoque une action que dans le cas où l'utilisateur utilise la souris. Or, de nombreuses personnes utilisent plutôt le clavier pour saisir des données dans un formulaire.

8.2 Utiliser les événements

C'est simple, il suffit de définir la propriété voulue de la balise concernée. En général, on appelle une fonction.

Exemple :

```
<p onmouseover="alert('Vous survolez ce texte !')">Un bout de texte</p>
```

9 utilisation du javascript dans le Html

Le javaScript doit être déclaré entre les balises

```
<script type="text/javascript">

    */ exemple de code java script avec utilisation fonction et log dans la console */
    tableMulitplication(5);

    var maVariable = 5 ;

    console.log(maVariable) ;

</script>
```

Deux façons de travailler sont possibles :

- déclarer le code la ou l'on en a besoin
  OU
- déclarer le code dans le Header dans une fonction et l'utiliser plus tard dans le code
  Exemple de fonction :

```
function mulitplication(nb1, nb2){
    var result = nb1*nb2;
    return result;
}
```

Il est aussi possible d'**externaliser votre code javaScript** dans un fichier .js et déclarer son utilisation dans le fichier Html (comme pour le fichier css) dans le Header de la façon suivantes :

```
<script type="text/javascript" src="fonctions.js"></script>
```