Introduction

1. Définition de DOM

DOM (Modèle Objet de Document ou *Document Object Model*) est le modèle d'accès aux différentes composantes des documents HTML ou encore XML.

Dans le chapitre Exploration de flux XML via DOM, nous verrons dans le détail comment exploiter des flux XML dans des scripts JavaScript.

Par son intermédiaire, vous disposerez d'une description structurée du script HTML et DOM fournit aussi le mode d'accès aux éléments constitutifs du modèle.

En tant que développeur, ce sera un allié précieux pour accéder et manipuler ces éléments depuis un langage de programmation (JavaScript par exemple mais pas seulement).

DOM va être pour nous un schéma d'accès pour atteindre les constituants (formulaire, champ...) d'une page Web balisée.

Ce modèle fournit une hiérarchie de programmation agençant les propriétés, les méthodes, les événements.

L'étude et la bonne compréhension du modèle DOM sont bien évidemment indispensables pour pouvoir utiliser au mieux les principales bibliothèques JavaScript (librairies), comme Prototype, jQuery, MooTools...

Le modèle DOM est un arbre constitué d'objets Node (nœuds). Il s'agit d'une structure arborescente avec un nœud de type racine et des nœuds enfants. Dans notre exemple ci-après (script HTML), le nœud `head` est un enfant du nœud `<html>`. Le nœud `<head>` est lui-même parent du nœud `<title>`. Les balises `<input>` intégrées dans la section `<body> ... </body>` représentent un exemple de nœuds de même niveau qui ont un même parent, le formulaire nommé `formulaire_saisie`.

```
<html>
    <head>
        <title>
        </title>
    </head>
    <body>
        <form name="formulaire_saisie">
            <input ... </input>
            <input ... </input>
        </form>
    </body>
</html>
```

Le modèle DOM est bien sûr applicable à tout document de type XML.

2. Définition de l'arborescence

Notion de nœud

Un langage de marquage comme HTML ou tout autre langage basé sur XML peut être schématisé comme une arborescence hiérarchisée. Les différentes composantes d'une telle arborescence sont désignées comme étant des nœuds. L'objet central du modèle DOM est pour cette raison l'objet Node. Les autres nœuds en dépendent.

Interface Node

L'interface Node va être notre boîte à outils par parcourir les différents niveaux de l'arborescence DOM. Elle est constituée de propriétés et de méthodes pour assurer les déplacements dans l'arbre.

Les propriétés (classées par ordre alphabétique)

- `attributes` : liste des attributs du nœud en cours.
- `childNodes` : liste des nœuds enfants.
- `firstChild` : premier enfant dans la filiation d'un nœud.
- `lastChild` : dernier enfant dans la filiation d'un nœud.
- `nextSibling` : prochain nœud d'un même type.
- `nodeName` : nom du nœud.
- `nodeType` : type du nœud.
- `nodeValue` : valeur contenue dans le nœud.
- `parentNode` : nœud parent.
- `previousSibling` : nœud précédent d'un type précisé.

Les méthodes (classées par ordre alphabétique)

- `appendChild()` : ajout d'un nœud enfant.
- `cloneNode()` : duplication d'un nœud.
- `deleteData()` : effacement des données en caractères.
- `getAttribute()` : recherche de la valeur d'un nœud attribut.
- `getAttributeNode()` : recherche d'un nœud attribut.
- `hasChildNodes()` : contrôle d'existence de nœuds enfants.
- `insertBefore()` : insertion d'un nœud.
- `removeAttribute()` : effacement de la valeur d'un nœud attribut.
- `removeAttributeNode()` : effacement d'un nœud attribut.
- `removeChild()` : suppression d'un nœud.
- `replaceChild()` : remplacement d'un nœud enfant.
- `setAttribute()` : définition de la valeur d'un nœud attribut.
- `setAttributeNode()` : définition d'un nœud attribut.

L'objet `document` est la racine de tous les nœuds d'un script Web. Cet objet particulier a ses propres méthodes s'appliquant au document dans sa globalité :

- `createElement` : création d'un nouvel élément pour l'arborescence.

- `createTextNode` : création d'un nœud de type texte.
- `createAttribut` : création d'un nœud de type attribut.
- `getElementById` : récupération de l'élément portant l'`id` spécifié (déjà largement étudié dans les exemples précédents du livre).
- `getElementsByName` : récupération de l'élément portant le nom spécifié.
- `getElementsByTagName` : récupération de l'élément ayant le tag spécifié.

Exemple

Soit un document Web comportant un tag `div` avec l'attribut `id` égal à `'container'`, essayons de placer à l'intérieur un autre div possédant le texte `'Bonjour le monde'` :

```
// Recherche de l'élément div ayant comme identifiant container
var ancienContainer = document.getElementById('container');

// Création en mémoire d'un élément div
var nouvelleDiv = document.createElement('div');

// Création du node de type texte
var texteDiv = document.createTextNode('Bonjour le monde');

// Affectation du node texteDiv au node nouvelleDiv
nouvelleDiv.appendChild(texteDiv);

// Intégration du nouveau div
ancienContainer.appendChild(nouvelleDiv);
```