Lycée LOUIS JOUVET Accompagner pour réussir	**Langage JAVASCRIPT** *DOM : Document Object Model*	***BTS CIEL*** ***Semestre 1*** ***2024_2025***

Lorsqu'on demande à un navigateur d'afficher une page Web, celui-ci va automatiquement créer un modèle objet de la page ou du document. Ce modèle objet correspond à une autre représentation de la page sous forme d'arborescence contenant des objets qui sont de type Node (nœuds).

Les navigateurs utilisent eux-mêmes cette arborescence qui va s'avérer très pratique à manipuler pour eux et notamment pour appliquer les styles aux bons éléments. Nous allons également pouvoir utiliser ce modèle objet en utilisant le langage JavaScript.

Exemple :

```
<!DOCTYPE html>
<html>
  <head>
    <title>Cours JavaScript</title>
    <meta charset="utf-8">
  </head>
  <body>
    <h1> Cours DOM</h1>
    <p>1er chapitre</p>
  </body>

</html>
```

Les objet nœuds du DOM

Le terme "nœud" est un terme générique qui sert à désigner tous les objets contenus dans le DOM. A l'extrémité de chaque branche du DOM se trouve un nœud.

A partir du nœud racine qui est le nœud HTML, 2 branches se forment : une première qui va aboutir au nœud HEAD et une deuxième qui aboutit à un nœud BODY.

De nouvelles branches se créent ensuite à partir des nœuds HEAD et BODY.

Des nœuds texte apparaissent pour deux raisons : soit parce qu'un élément contient effectivement du texte, soit parce qu'il y a eu un retour à la ligne ou parce qu'il y a un espace entre deux éléments contenus dans l'élément html (aucun nœud de type texte n'est créé entre les balises ouvrantes de html et de head, ni entre les balises fermantes de body et de html).

Un caractère spécial va nous indiquer si un nœud de type texte a été constitué par une nouvelle ligne (caractère ↲), un espace (caractère ␣) ou du texte (caractère #).

Une autre représentation du DOM peut être obtenue en inspectant la page. Dans cette représentation, certains navigateurs comme Chrome ne mentionnent pas les nœuds texte crées par des espaces ou des retours à la ligne dans le code car ils savent que ce ne sont que des nœuds "esthétiques" et non utiles au code.

Les types de nœuds du DOM

On peut distinguer les nœuds les uns des autres en fonction de s'il s'agit d'un nœud constitué par un texte, par un élément, par un commentaire, etc. On peut utiliser des propriétés et des méthodes différentes avec chaque type de nœud puisqu'ils vont dépendre d'interfaces différentes.

Les différents types de nœuds :

Constante	Valeur	Description
ELEMENT_NODE	1	Représente un nœud élément (comme p ou divpar exemple)>
TEXT_NODE	3	Représente un nœud de type texte
PROCESSING_INSTRUCTION_NODE	7	Nœud valable dans le cas d'un document XML. Nous ne nous en préoccuperons pas ici.
COMMENT_NODE	8	Représente un nœud commentaire
DOCUMENT_NODE	9	Représente le nœud formé par le document en soi
DOCUMENT_TYPE_NODE	10	Représente le nœud doctype
DOCUMENT_FRAGMENT_NODE	11	Représente un objet document minimal qui n'a pas de parent (ce type de nœud ne nous intéressera pas ici

L'un des intérêts majeurs du DOM et des nœuds va être qu'on va pouvoir se déplacer de nœuds en nœuds pour manipuler des éléments en utilisant le JavaScript.

Les interfaces composant le DOM

Le DOM est un ensemble d'interfaces qui peuvent fonctionner ensemble et permettre d'accéder à et de manipuler divers éléments des documents en JavaScript.

Quelques exemples d'interfaces du DOM :

- L'interface windows qu'on a déjà étudié et qui est liée au DOM ;
- L'interface Event qui représente tout événement qui a lieu dans le DOM ;
- L'interface EventTarget qui est une interface que vont implémenter les objets qui peuvent recevoir des évènements ;
- L'interface Node qui est l'interface de base pour une grande partie des objets du DOM ;
- L'interface Document qui représente le document actuel et qui va être l'interface la plus utilisée ;
- L'interface Element qui est l'interface de base pour tous les objets d'un document ;

Le JavaScript est un langage à héritage simple. Cela signifie qu'une interface ne peut hériter que d'une seule autre interface. Les mixin sont des sortes d'interfaces qui permettent de contourner cette limitation : une interface ne pourra hériter que d'une autre interface mais pourra également implémenter plusieurs mixin.

Accès aux éléments à partir de l'ensemble du document

L'objet document est un objet important, qui représente l'ensemble de l'arborescence du document analysé. Il possède de nombreuses propriétés et méthodes. Cinq de ces dernières permettent de "pointer" directement un ou plusieurs éléments dans le document:

La méthode getElementById
Elle permet de sélectionner un élément d'identifiant donné dans une page.
Exemple : si on a dans la page ** (...) **, **document.getElementById("intro")** permettra de sélectionner précisément l'élément p en question.

La méthode getElementsByName
Elle permet de sélectionner les éléments portant un nom donné dans une page ; mais cette méthode n'est pas supportée par Internet Explorer ;

La méthode getElementsByTagName
Elle permet de sélectionner les éléments portant un nom de balise donné dans une page.

La méthode querySelector
Elle prend en argument une expression CSS, permet de cibler directement le *premier* élément d'un ensemble de nœuds.
Exemple : **document.querySelector("#snir");** permet de sélectionner l'élément d'identifiant snir,
mais **document.querySelector(".bts");** ne sélectionne que le premier élément de classe bts.

La méthode querySelectorAll
Elle prend en argument une expression CSS et permet de cibler tous les éléments d'un ensemble de nœuds.
Exemple : **document.querySelectorAll("#snir");** renvoie un tableau d'un item contenant l'élément d'identifiant snir, et **document.querySelectorAll(".bts");** renvoie un tableau contenant tous les éléments de classe bts.

On peut également accéder aux divers éléments d'un document par leur numéro d'ordre dans ce dernier. Par exemple, les collections document.forms et document.images permettent d'accéder aux divers éléments de formulaires et images du document. Cet

Accès aux nœuds à partir d'un élément quelconque du document

Il existe aussi des méthodes qui, à partir d'un élément donné, permettent d'accéder à certains de ses descendants :

La méthode getElementsByTagName()
Elle permet d'obtenir une collection d'éléments *descendants* de l'élément courant, et possédant un nom de balise donné.
Exemple :

```
parag1=document.getElementById('monpar') ;
liens=parag1.getElementsByTagName('a') ;
```

liens renvoie la collection de liens dans le paragraphe identifié par monpar.

La méthode getAttribute(nom_d_attribut)
Elle permet de sélectionner un attribut particulier d'un élément déterminé.

Accès relatif

Les méthodes précédentes demandent de connaître précisément soit l'identifiant, soit le nom exact du nœud cherché. Mais il est possible d'accéder à des collections de nœuds dont on ne connaît pas ces caractéristiques à priori.
Dans toute la suite, elt désignera un élément que l'on aura au préalable identifié (par exemple par une méthode getElementById).

Accès aux attributs

Contrairement à la méthode getAttribute(), qui permet de retourner la valeur d'un attribut de nom donné, la propriété attributes renvoie à la collection complète des attributs d'un élément. Par exemple, si elt désigne l'élément img suivant, cette méthode renverra une liste constituée des nœuds src, alt, width et height (dans cet ordre) :

```
<img src="oiseau.gif" alt="Un oiseau sur un arbre" width="45" height="60">
```

Accès aux nœuds enfants

On peut accéder très facilement à la liste des nœuds-enfants d'un élément donné. Pour cela, trois propriétés existent.

- elt.childNodes donne la liste de tous les nœuds-enfants de l'élément elt sous la forme d'un tableau.
- elt.firstChild est équivalent à elt.childNodes[0], et renvoie le premier nœud-enfant de l'élément elt.
- elt.lastChild renvoie le dernier nœud-enfant de l'élément elt.
- elt.lastElementChild renvoie le dernier élément-enfant de l'élément elt.

Accès aux nœuds-frères

Le nœud courant peut avoir des frères. On accède au frère précédent à l'aide de la propriété previousSibling, et au nœud suivant par la propriété nextSibling.

Accès aux nœuds ancêtres

On peut remonter d'un cran dans la hiérarchie des nœuds avec la propriété parentNode, et revenir à l'élément-racine d'un document (dans le cas d'un document HTML, il s'agit de html), avec la propriété documentElement.

Accès par les collections

Quatre collections donnent accès à différents éléments du document :

- window.frames permet d'accéder à la liste des *frames* du document courant ; chaque élément de cette collection est alors du type window, et manipulable grâce aux propriétés et méthodes correspondantes (voir le chapitre dédié à la manipulation des fenêtres du navigateur).
- document.forms, document.images et document.links sont les collections des éléments de formulaire, d'images et de liens du document.

Obtenir des informations sur la nature d'un nœud

Deux propriétés permettent d'obtenir des informations sur la nature d'un nœud déterminé : nodeName et nodeType.

- nodeName renvoie le nom du nœud courant. Par exemple, si elt désigne un élément img, elt.nodeName renvoie la chaîne de caractères "img".
- nodeType est d'un usage moins direct. Cette propriété renvoie au type de nœud du nœud courant, selon un code numérique.

Obtenir des informations sur le contenu d'un nœud

Deux méthodes renvoient des booléens permettant de déterminer si un nœud donné possède des enfants, et/ou des attributs. Ces méthodes, aux noms explicites, sont respectivement hasChildNodes() et hasAttributes().