MANIPULER LE CODE HTML (PARTIE 1/2)

Le Document Object Model

Le **Document Object Model** (abrégé **DOM**) est une interface de programmation pour les documents XML et HTML.

Une interface de programmation, qu'on appelle aussi une API (pour Application Programming Interface), est un ensemble d'outils qui permettent de faire communiquer entre eux plusieurs programmes ou, dans le cas présent, différents langages. Le terme API reviendra souvent, quel que soit le langage de programmation que vous apprendrez.

**getElementById()**

Cette méthode permet d'accéder à un élément en connaissant son ID qui est simplement l'attribut id de l'élément.

**getElementsByTagName()**

Cette méthode permet de récupérer, sous la forme d'un tableau, tous les éléments de la famille.

**getElementsByName()**

Cette méthode est semblable à getElementsByTagName() et permet de ne récupérer que les éléments qui possèdent un attribut name que vous spécifiez. L'attribut name n'est utilisé qu'au sein des formulaires, et est déprécié depuis la spécification HTML5 dans tout autre élément que celui d'un formulaire. Par exemple, vous pouvez vous en servir pour un élément <input> mais pas pour un élément <map>.

Sachez aussi que cette méthode est dépréciée en XHTML mais est standardisée en HTML5.

**L'héritage des propriétés et des méthodes**

Le JavaScript voit les éléments HTML comme étant des objets, cela veut donc dire que chaque élément HTML possède des propriétés et des méthodes. Cependant faites bien attention parce que tous ne possèdent pas les mêmes propriétés et méthodes. Certaines sont néanmoins communes à tous les éléments HTML, car tous les éléments HTML sont d'un même type : le type Node, qui signifie « nœud » en anglais.

**Notion d'héritage**

Nous avons vu qu'un élément <div> est un objet HTMLDivElement, mais un objet, en JavaScript, peut appartenir à différents groupes. Ainsi, notre <div> est un HTMLDivElement, qui est un sous-objet d'HTMLElement qui est lui-même un sous-objet d'Element. Element est enfin un sous-objet de Node. Ce schéma est plus parlant :

En JavaScript, un objet peut appartenir à plusieurs groupes

L'objet **Node** apporte un certain nombre de propriétés et de méthodes qui pourront être utilisées depuis un de ses sous-objets. En clair, les sous-objets *héritent* des propriétés et méthodes de leurs objets parents. Voilà donc ce que l'on appelle l'**héritage**.

**En résumé**

* Le DOM va servir à accéder aux éléments HTML présents dans un document afin de les modifier et d'interagir avec eux.
* L'objet ***window***est un objet global qui représente la fenêtre du navigateur. ***document***, quant à lui, est un sous-objet de ***window***et représente la page Web. C'est grâce à lui que l'on va pouvoir accéder aux éléments HTML de la page Web.
* Les éléments de la page sont structurés comme un arbre généalogique, avec l'élément <html>comme élément fondateur.
* Différentes méthodes, comme getElementById(), getElementsByTagName(),querySelector() ou querySelectorAll(), sont disponibles pour accéder aux éléments.
* Les attributs peuvent tous être modifiés grâce à setAttribute(). Certains éléments possèdent des propriétés qui permettent de modifier ces attributs.
* La propriété innerHTML permet de récupérer ou de définir le code HTML présent à l'intérieur d'un élément.
* De leur côté, textContent et innerText ne sont capables que de définir ou récupérer du texte brut, sans aucunes balises HTML.