## Table des matières

# Chapitre 3 : DOM (Document Object Model)

## I. Introduction

Quand une page Web est chargée, le navigateur crée un modèle objet de document (Document Object Model) de la page. Le modèle DOM HTML est construit comme un arbre d'objets.

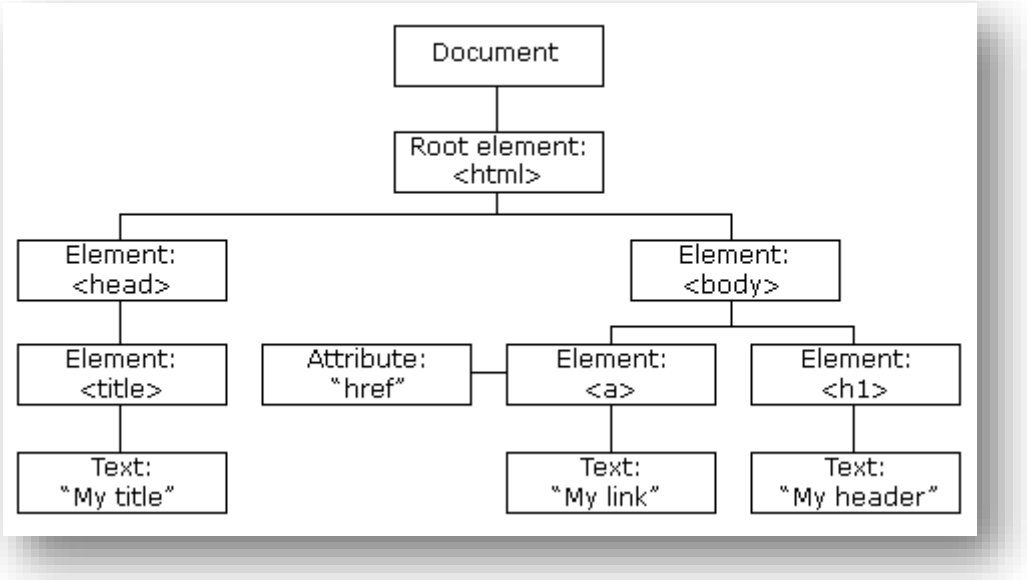



*Figure 1 : Arbre d'objets DOM*

Avec le modèle objet, JavaScript obtient toute la puissance dont il a besoin pour créer HTML dynamique :

- JavaScript peut changer tous les éléments HTML dans la page
- JavaScript peut changer tous les attributs HTML dans la page
- JavaScript peut changer tous les styles CSS dans la page
- JavaScript peut supprimer des éléments et attributs HTML existants
- JavaScript peut ajouter de nouveaux éléments et attributs HTML
- JavaScript peut réagir à tous les événements de HTML existants dans la page
- JavaScript peut créer de nouveaux événements HTML dans la page

## II. Qu'est-ce que le DOM ?

Le DOM est un W3C (World Wide Web Consortium) standard. Le DOM définit une norme pour l'accès aux documents, La norme DOM W3C est séparée en trois parties distinctes :

- Noyau DOM - modèle standard pour tous les types de documents
- DOM XML - modèle standard pour les documents XML
- DOM HTML - modèle standard pour les documents HTML

## 1. Qu'est-ce que le DOM de HTML

Le code HTML DOM est une interface de programmation objet et un modèle standard pour HTML. Il définit :

- Les éléments HTML comme des objets
- Les propriétés de l'ensemble des éléments HTML
- Les méthodes pour accéder à tous les éléments HTML
- Les événements pour tous les éléments HTML

En d'autres termes : Le DOM HTML est un standard pour savoir comment obtenir, modifier, ajouter ou supprimer des éléments HTML

## 2. L'interface de programmation DOM

L'interface de programmation est les propriétés et méthodes de chaque objet.

- Une propriété est une valeur que vous pouvez obtenir ou définir (comme changer le contenu d'un élément HTML).
- Une méthode est une action que vous pouvez faire (comme ajouter ou de supprimer un élément HTML)

***Exemple :***

```
<html>
<body>
        <p id="intro">Hello World!</p>
        <script>
                var txt=document.getElementById("intro").innerHTML;
                document.write(txt);
        </script>
</body>
</html>
```

Dans l'exemple ci-dessus, ***getElementById*** est une méthode, tandis qu'***innerHTML*** est une propriété.

Dans l'exemple ci-dessus la méthode utilisée ***getElementById("intro")*** pour trouver l'élément dont l'id est « *intro* ».

La propriété ***innerHTML*** est utile pour obtenir ou remplacer le contenu des éléments HTML.

# III. L'objet Document

Dans le modèle d'objet DOM HTML, l'objet document représente votre page web. L'objet document est le propriétaire de tous les autres objets dans votre page Web. Si vous voulez accéder à des objets dans une page HTML, vous commencez toujours par accéder à l'objet document.

| Exemples | |
|---|---|
| **document.getElementById ()** | *Trouver des éléments par nom de balise* |
| **document.getElementsByClassName ()** | *Trouver des éléments par les objets d'élément HTML* |

## 1. Trouver les éléments HTML

Souvent, avec JavaScript, vous voulez manipuler des éléments HTML.

Pour ce faire, vous devez trouver les premiers éléments. Il y a plusieurs façons de le faire :

### 1.1. Trouver des éléments HTML par identifiant

La façon la plus simple de trouver des éléments HTML dans le DOM, est d'utiliser l'élément **id** et la méthode ***getElementById*** de l'objet document.

***Exemple :*** Cet exemple recherche l'élément avec **id = "intro"**

```
var x=document.getElementById("intro");
```

### 1.2. Trouver des éléments HTML par nom de balise

La fonction utilisée est ***getElementsByTagName("Nom de balise")***.

***Exemple :*** Cet exemple recherche l'élément avec **id = "main"**, puis trouve tous les éléments **<p>** imbriqués dans la balise "**main**"

```
var x=document.getElementById("main");
var y=x.getElementsByTagName("p");
```

### 1.3. Trouver des éléments HTML par nom de classe

La fonction utilisée est ***getElementsByClassName("Nom de classe")***.

***Exemple :*** Cet exemple cherche la classe avec le nom « **intro** »

```
var x= getElementsByClassName ("intro");
```

### 1.4. Trouver des éléments HTML par des collections d'objets HTML

Cet exemple recherche l'élément de formulaire avec **id = "frm1"**, dans la collection de formulaires, et affiche toutes les valeurs d'éléments du formulaire en utilisant la propriété ***elements***.

***Exemple :***

```
<!DOCTYPE html>
<html>
<body>
  <form id="frm1" action="form_action.asp">
          First name: <input type="text" name="fname" value="Donald"><br>
          Last name: <input type="text" name="lname" value="Duck"><br>
          <input type="submit" value="Submit">
  </form>
  <p>Click the "Try it" button to return the value of each element in the form.</p>
  <p id="demo"></p>
  <button onclick="myFunction()">Try it</button>
  <script>
```

```
        function myFunction()  {
                var x = document.getElementById("frm1");
                var txt = "";
                for (var i=0;i<x.length;i++)  {
                        txt = txt + x.elements[i].value + "<br>";
                 }
                document.getElementById("demo").innerHTML=txt;
        }
  </script>
</body>
</html>
```

Les collections d'objets HTML suivantes sont accessibles :

- document.anchors
- document.body
- document.documentElement
- document.embeds
- document.forms
- document.head
- document.images
- document.links
- document.scripts
- document.title

## 2. Changer des éléments HTML

Le DOM HTML JavaScript permet de modifier le contenu des éléments HTML.

### 2.1. Changer OutputStream

La méthode ***write*** de l'objet document permet d'écrire dans une page web.

***Exemple :***

```
<html>
<body>
        <script>
                document.write(Date());
        </script>
</body>
</html>
```

### 2.2. Changer le contenu HTML

La meilleure façon de modifier le contenu d'un élément HTML est d'utiliser la propriété **innerHTML**.

Pour modifier le contenu d'un élément HTML, utilisez la syntaxe suivante :

***document.getElementById(id).innerHTML*** **= new HTML**

***Exemple 1 :*** Changer le contenu de l'élément **<p>**

```
<!DOCTYPE html>
<html>
<body>
        <p id="p1">Hello World!</p>
 <script>
        document.getElementById("p1").innerHTML="New text!";
 </script>
</body>
</html>
```

***Exemple 2 :*** Changer le contenu de l'élément **<h1>**

```
        <h1 id="header">Old Header</h1>
 <script>
        var element=document.getElementById("header");
        element.innerHTML="New Header";
 </script>
```

### 2.3. Changer la valeur d'un attribut

Pour modifier la valeur d'un attribut HTML, utilisez la syntaxe suivante :

```
document.getElementById(id).attribute = new value
```

***Exemple :*** modifier la valeur de l'attribut **src** d'un élément **<img>**

```
 <img id="image" src="smiley.gif">
 <script>
        document.getElementById("image").src = "landscape.jpg";
 </script>
>
```

### 2.4. Changer le style HTML

Le DOM permet de changer le style des éléments HTML. Pour changer le style d'un élément HTML, utilisez la syntaxe suivante :

```
document.getElementById(id).style.property = new style
```

***Exemple :*** modifier le style d'un élément **<p>**

```
<!DOCTYPE html>
<html>
<body>
```

```
        <p id="p2">Hello World!</p>
  <script>
        document.getElementById("p2").style.color = "blue";
  </script>
        <p>The paragraph above was changed by a script.</p>
</body>
</html>
```

Le DOM HTML vous permet d'exécuter du code lorsqu'un événement se produit.

***Exemple :*** modifier le style de l'élément HTML avec id = "id1", lorsque l'utilisateur clique sur un bouton.

```
<!DOCTYPE html>
<html>
<body>
        <h1 id="id1">My Heading 1</h1>
        <button type="button" onclick="document.getElementById('id1').style.color='red'">
                Click Me!
        </button>
</body>
</html>
```

## 3. Les événements et DOM

DOM HTML permet aux JavaScripts de réagir aux événements de HTML en changeant un contenu HTML

### 3.1. Changement d'un contenu HTML en utilisant les événements

Un script JavaScript peut être exécuté lorsqu'un événement se produit, comme quand un utilisateur clique sur un élément HTML.

Pour exécuter du code lorsque l'utilisateur clique sur un élément, ajouter du code JavaScript à un attribut d'événement HTML

***Exemple 1 :*** le contenu de l'élément **<h1>** est changé lorsqu'un utilisateur clique sur **<h1>**.

```
<!DOCTYPE html>
<html>
<body>
        <h1 onclick="this.innerHTML='Ooops!'">Click on this text!</h1>
</body>
</html>
```

***Exemple 2 :*** une fonction est appelée depuis le gestionnaire d'événements, la fonction permet de changer le contenu d'un élément dont l'id est saisi en paramètre.

```
<!DOCTYPE html>
<html>
<head>
  <script>
        function changetext(id) {
                id.innerHTML="Ooops!";
        }
  </script>
</head>
<body>
        <h1 onclick="changetext(this)">Click on this text!</h1>
</body>
</html>
```

### 3.2. Attribuer des événements en utilisant le DOM HTML

Le DOM HTML vous permet d'assigner des événements aux éléments HTML utilisant JavaScript.

***Exemple :*** Attribuer un événement *onclick* à un élément bouton.

```
<script>
        document.getElementById("myBtn").onclick=function(){ displayDate() };
</script>
```

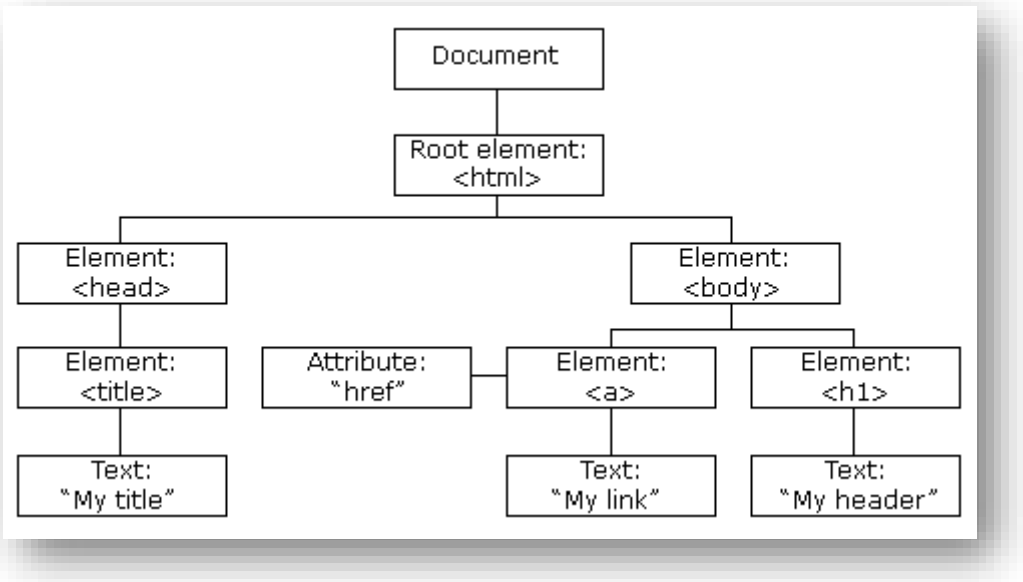
## 4. Navigation dans un document en utilisant DOM

Avec le DOM HTML, vous pouvez naviguer dans l'arborescence de nœuds à l'aide des relations entre les nœuds.

### 4.1. Nœuds DOM

Conformément à la norme W3C HTML DOM, tout en un document HTML est un nœud :

- La totalité du document est un nœud document
- Chaque élément HTML est un nœud élément
- Le texte à l'intérieur des éléments HTML sont des nœuds texte
- Chaque attribut HTML est un nœud attribut
- Tous les commentaires sont les nœuds commentaire

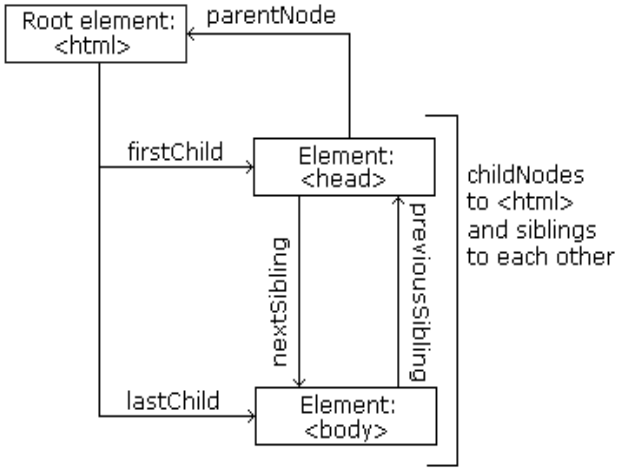



Avec le DOM HTML, tous les nœuds de l'arborescence de nœuds peuvent être accessibles par JavaScript.

Les nouveaux nœuds peuvent être créés, et tous les nœuds peuvent être modifiés ou supprimés.

### 4.2. Relations entre les nœuds

```
<html>
 <head>
   <title>DOM Tutorial</title>
</head>
 <body>
   <h1>DOM Lesson one</h1>
   <p>Hello world!</p>
 </body>
</html>
```




Vous pouvez utiliser les propriétés de nœuds suivants pour naviguer entre les nœuds avec JavaScript :

- parentNode
- childNodes[nodenumber]
- firstChild
- lastChild
- nextSibling
- previousSibling

### 4.3. ChildNodes et NodeValue

En plus de la propriété **innerHTML**, vous pouvez également utiliser les propriétés ***childNodes*** et ***nodeValue*** pour obtenir le contenu d'un élément.

***Exemple :*** Le code suivant obtient la valeur de l'élément **<p>** avec **id = "intro"**.

```
<html>
<body>
        <p id="intro">Hello World!</p>
  <script>
        var txt=document.getElementById("intro").childNodes[0].nodeValue;
        document.write(txt);
  </script>
</body>
</html>
```

### 4.4. Nœud ROOT

Il y a deux propriétés particulières qui permettent d'accéder à l'intégralité du document :

- ***document.documentElement*** : Le document complet
- ***document.body*** : Le corps du document

***Exemple :***

```
<p>Hello World!</p>
<div>
        <p>The DOM is very useful!</p>
        <p>This example demonstrates the <b>document.body</b> property.</p>
</div>
  <script>
        alert(document.body.innerHTML);
  </script>
```

### 4.5. La propriété nodeName

La propriété ***nodeName*** spécifie le nom d'un nœud :

- *nodeName est en lecture seule*
- *nodeName d'un nœud d'élément est le même que le nom de la balise*
- *nodeName d'un nœud d'attribut est le nom de l'attribut*
- *nodeName d'un nœud de texte est toujours # texte*
- *nodeName du nœud de document est toujours # document*

***Remarque :*** *nodeName* contient toujours le nom de la balise majuscule d'un élément HTML.

### 4.6. La propriété nodeValue

La propriété ***nodeValue*** spécifie la valeur d'un nœud :

- *nodeValue pour les nœuds d'élément est indéfini*
- *nodeValue pour les nœuds de texte est le texte lui-même*
- *nodeValue pour les nœuds d'attribut est la valeur de l'attribut*

### 4.7. La propriété nodeType

La propriété ***nodeType*** retourne le type de nœud. ***nodeType*** est en lecture seule.

Les types de nœuds les plus importants sont :

| Element type | NodeType |
|---|---|
| **Element** | 1 |
| **Attribute** | 2 |
| **Text** | 3 |
| **Comment** | 8 |
| **Document** | 9 |

## 5. Manipulation des nœuds DOM

### 5.1. Ajouter un nouveau nœud HTML

Pour ajouter un nouvel élément au DOM HTML, vous devez créer l'élément (nœud d'élément) d'abord, puis l'ajouter à un élément existant.

***Exemple 1 :***

```
<body>
        <div id="div1">
                <p id="p1">This is a paragraph.</p>
                <p id="p2">This is another paragraph.</p>
        </div>
 <script>
        var para=document.createElement("p");
        var node=document.createTextNode("This is new.");
        para.appendChild(node);
        var element=document.getElementById("div1");
        element.appendChild(para);
 </script>
</body>
```

Avec la méthode *appendChild()* dans l'exemple précédent, le nouvel élément est ajouté au dernier enfant du parent.

Si vous ne voulez pas, vous pouvez utiliser la méthode *insertBefore().*

***Exemple 2 :***

```
<body>
        <div id="div1">
                <p id="p1">This is a paragraph.</p>
                <p id="p2">This is another paragraph.</p>
        </div>
 <script>
        var para=document.createElement("p");
        var node=document.createTextNode("This is new.");
        para.appendChild(node);
        var element=document.getElementById("div1");
        element.insertBefore(para,child);
 </script>
</body>
```

### 5.2. Supprimer un nœud HTML

Pour supprimer un élément HTML, vous devez connaître le parent de l'élément.

***Exemple :***

```
<body>
        <div id="div1">
                <p id="p1">This is a paragraph.</p>
                <p id="p2">This is another paragraph.</p>
        </div>
 <script>
        var parent=document.getElementById("div1");
        var child=document.getElementById("p1");
        parent.removeChild(child);
 </script>
</body>
```

### 5.3. Remplacer un nœud HTML

Pour remplacer un élément de la DOM HTML, utilisez la *replaceChild* (méthode).

***Exemple :***

```
<body>
        <div id="div1">
                <p id="p1">This is a paragraph.</p>
                <p id="p2">This is another paragraph.</p>
        </div>
 <script>
        var para=document.createElement("p");
        var node=document.createTextNode("This is new.");
        para.appendChild(node);

        var parent=document.getElementById("div1");
        var child=document.getElementById("p1");
        parent.replaceChild(para,child);
 </script>
</body>
```

## 6. DOM nodeList

### 6.1. Accès à une nodeList

La méthode *getElementsByTagName()* retourne une liste de nœuds. Une liste de nœuds est un tableau de nœuds.

***Exemple :*** Le code suivant sélectionne tous les nœuds **<p>** dans un document

```
var x=document.getElementsByTagName("p");
```

Pour accéder au deuxième élément (les éléments sont indexés à partir de 0) :

```
y=x[1] ;
```

### 6.2. Longueur d'une nodeList

La longueur d'un *nodeList* est donnée par la propriété *length* comme suit.

***Exemple :*** Le code suivant sélectionne tous les nœuds **<p>** dans un document

```
x=document.getElementsByTagName("p");
for (i=0; i<x.length; i++) {
        document.write(x[i].innerHTML);
        document.write("<br />");
}
```

# 7. L'objet Window

## 7.1. Présentation

L'objet *Window* est supporté par tous les navigateurs. Il représente la fenêtre du navigateur.

Tous les objets globaux JavaScript, les fonctions et les variables deviennent automatiquement membres de l'objet **window**. Les variables globales sont la propriété de l'objet *window*. Les fonctions globales sont des méthodes de l'objet window.

Même l'objet *document* (du DOM HTML) est une propriété de l'objet *Window.*

***Exemple :***

```
window.document.getElementById("header");
```

est même que

```
document.getElementById("header");
```

## 7.2. Méthodes de l'objet window

| Méthodes | Rôle |
|---|---|
| window.innerHeight | la hauteur intérieure de la fenêtre du navigateur |
| window.innerWidth | la largeur intérieure de la fenêtre du navigateur |
| window.open(String url [, String nom] [, String options] ) | Ouvrir une fenêtre |
| window.close() | Fermer une fenêtre |
| window.moveTo(x,y) | Déplacer une fenêtre ((x,y) est la position de coin supérieur gauche de la fenêtre) |
| window.resizeTo(largeur, hauteur) | Redimensionner une fenêtre |