Accéder aux objets

1. Par la méthode getElementById

La méthode JavaScript `getElementById` parcourt le document HTML à la recherche d'un nœud unique qui a été spécifié par l'attribut `id`. Cet identifiant `id` doit être unique dans le document.

Le terme *Element* de `getElementByID` est bien au singulier car il ne peut y avoir qu'un seul identifiant portant ce nom.

Exemple

Au clic sur un titre de niveau 1, retourner dans une boîte d'alerte le nom du nœud (*nodeName*).

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Le DOM</title>
<meta charset="UTF-8">
<script>
function valeur()  {
var x = document.getElementById("titre").nodeName;
alert(x);
}
</script>
<style>
h1 { cursor: pointer;}
</style>
</head>
<body>
<h1 id="titre" onclick="valeur()">Titre de niveau 1</h1>
<p>Cliquez sur le titre</p>
</body>
</html>
```

La méthode `getElementById`, appliquée au document, accède à l'identifiant repris en argument ("titre") et la propriété de nœud `nodeName` lui est appliquée. Cette valeur, stockée dans la variable x, est affichée dans la boîte d'alerte.

Cette méthode `getElementById` n'est que rarement utilisée dans un document XML car elle recherche les attributs de type id qui doivent alors être définis dans un DTD particulier.

2. Par la méthode getElementsByName

La méthode `getElementsByName` permet de sélectionner les éléments portant un nom donné, spécifié par l'attribut `name`. Les éléments portant le même nom sont stockés dans une liste de nœuds. Cette liste se gère comme un tableau `Array`.

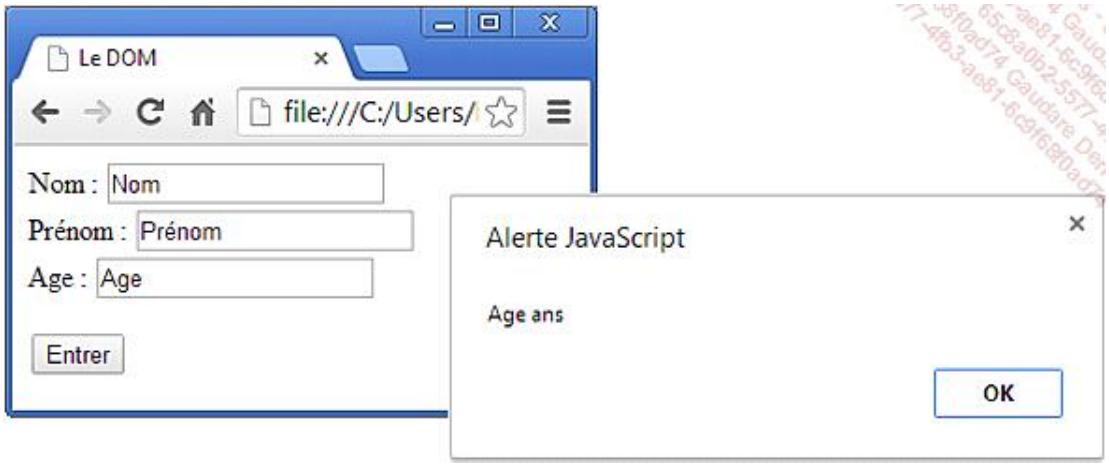
Le terme *Elements* de `getElementsByName` est bien au pluriel car plusieurs éléments portant le même nom peuvent se trouver dans le document.

Exemple

Accédons à la valeur de la troisième ligne de texte par la méthode `getElementsByName`.

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Le DOM</title>
<meta charset="UTF-8">
<script>
function test() {
var age = document.getElementsByName("in")[2].value;
alert(age + " ans");
}
</script>
</head>
<body>
<form>
Nom : <input type="text" name="in" value="Nom"><br>
Prénom : <input type="text" name="in" value="Prénom"><br>
Age : <input type="text" name="in" value="Age">
<p><input type="submit" value="Entrer" onclick="test()"></p>
</form>
</body>
</html>
```

La méthode `getElementsByName`, appliquée au document, accède à la troisième balise ayant l'attribut `name="in"` par `getElementsByName("in")[2]`. Sa valeur est récupérée par `value`. Cette valeur, stockée dans la variable `age`, est affichée dans la boîte d'alerte.



L'attribut name est un héritage du HTML 4.0. Son emploi est déprécié (deprecated) en HTML5 au profit de l'identifiant `id`. Ainsi l'emploi de `getElementsByName` est de moins en moins fréquent. Il est même ignoré dans certaines parutions récentes.

3. Par la méthode getElementsByTagName

La méthode `getElementsByTagName` parcourt le document à la recherche de toutes les balises d'un type spécifique, signalé en argument. Ces balises sont contenues dans une liste (*nodeList*) qui se gère comme les tableaux de type *Array*.

Le terme *Elements* de `getElementsByName` est bien au pluriel car il peut y avoir plusieurs balises de même type dans le document.

Exemple

Accédons à la valeur de la troisième ligne de texte par `getElementsByTagName`.

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Le DOM</title>
<meta charset="UTF-8">
<script>
function test() {
var age = document.getElementsByTagName("input")[2].value;
alert(age + " ans");
}
</script>
</head>
<body>
<form>
Nom : <input type="text" name="in" value="Nom"><br>
Prénom : <input type="text" name="in" value="Prénom"><br>
Age : <input type="text" name="in" value="Age">
<p><input type="submit" value="Entrer" onclick="test()"></p>
</form>
</body>
</html>
```

La méthode `getElementsByTagName`, appliquée au document, accède à la troisième balise `<input>` par `getElementsByTagName("input")[2]`. Sa valeur est récupérée par `value`. Cette valeur, stockée dans la variable `age`, est affichée dans la boîte d'alerte.

La capture est identique à celle du point précédent.

Cette méthode `getElementsByTagName` est très fréquemment utilisée dans les documents XML.

4. Par la méthode getElementsByClassName

La méthode `getElementsByClassName` renvoie un ou des éléments comportant le même nom de classe. Ces éléments sont contenus dans une liste (*nodelist*) qui se gère comme un *Array*.

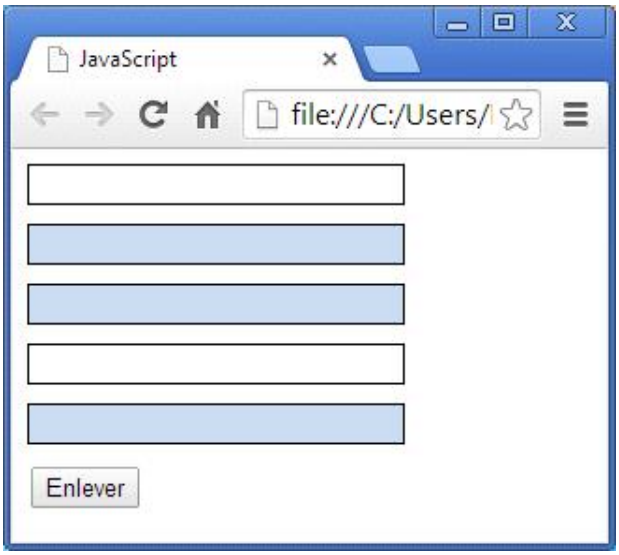
Le terme *Elements* de `getElementsByClassName` est bien au pluriel car il peut y avoir plusieurs balises possédant le même attribut `class`.

Reconnue par toutes les versions de Firefox et de Google Chrome, il a fallu attendre Internet Explorer 9 pour qu'elle soit prise en charge par Microsoft.

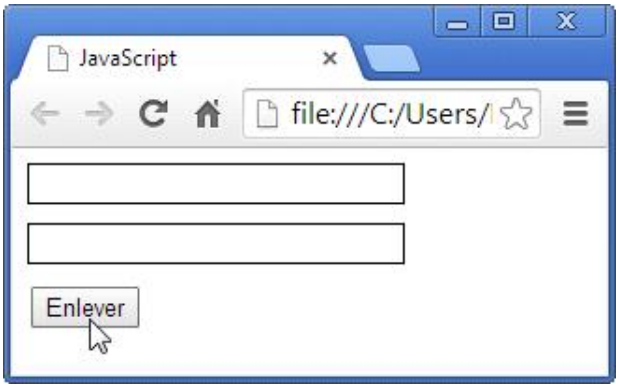
Exemple

Au clic sur un bouton, faisons disparaître toutes les divisions colorées, soit celles avec la classe `couleur`.

Situation initiale :



Au clic sur le bouton :




```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>JavaScript</title>
<meta charset="UTF-8">
<style>
div { width:200px;height: 20px;
      border: 1px solid black;
      margin-bottom: 10px;}
.couleur { background: rgb(195,215,235);}
</style>
</head>
<body>
<div></div>
<div class="couleur"></div>
<div class="couleur"></div>
<div></div>
<div class="couleur"></div>
<button type="button" onclick="colorier()">Enlever</button>
<script>
function colorier(){
var elements = document.getElementsByClassName("couleur");
for(var i = 0, length = elements.length; i < length; i++) {
elements[i].style.display = 'none';
}
}
</script>
</body>
</html>
```

5. Par les propriétés des nœuds

Il est théoriquement possible d'accéder à n'importe quel élément par un code du genre :

```
x.parentNode.lastChild.childNodes[2].firstChild.nextSibling;
```

Cette façon de procéder ne se révèle cependant pas très pratique à l'usage car :

- Le code devient rapidement illisible.

- Une simple mise à jour de la page risque de modifier complètement l'arborescence du document et nécessiterait alors la réécriture complète du code.
- Les navigateurs n'ont pas tous la même interprétation du DOM.

Ainsi, les développeurs privilégient les méthodes `getElementById`, `getElementsByName` ou `getElementsByTagName` pour se rapprocher de l'élément et, à partir de là, utilisent les propriétés `firstChild`, `parentNode` ou autres propriétés similaires pour accéder à l'élément souhaité.

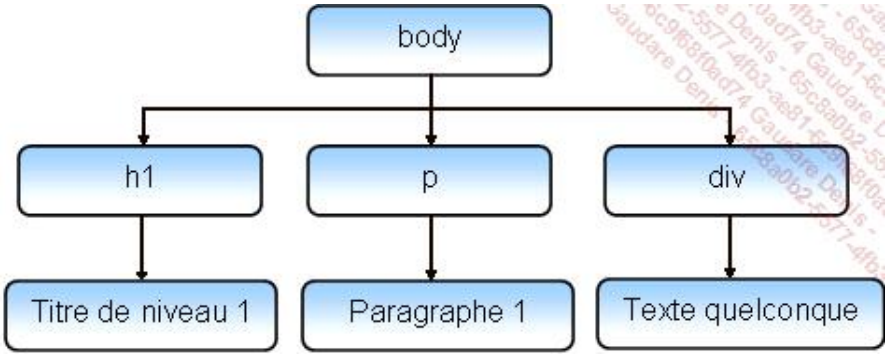
Mettons en pratique cette méthode, c'est-à-dire un saut de longue distance suivi d'une exploration limitée.

Exemple 1

Soit le code suivant :

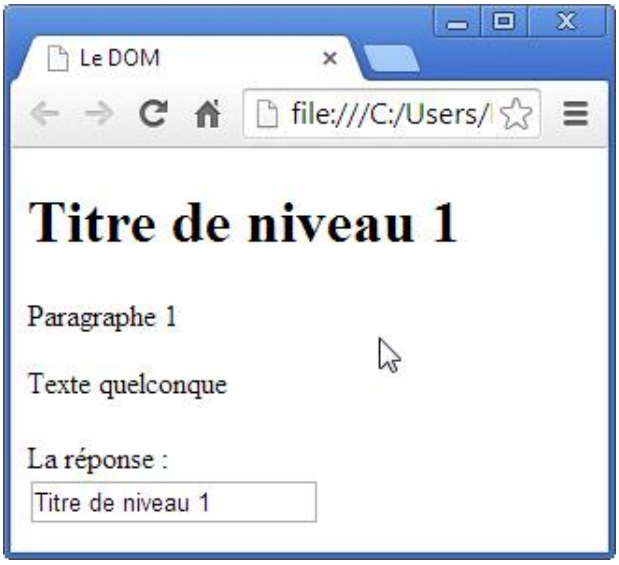
```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Le DOM</title>
<meta charset="UTF-8">
</head>
<body>
<h1 id="titre">Titre de niveau 1</h1>
<p>Paragraphe 1</p>
<div>Texte quelconque</div>
<form>
La réponse :<br>
<input id="texte" type="text">
</form>
</body>
</html>
```

L'arborescence se présente comme suit :



Au clic sur le document, affichons dans une ligne de texte le texte compris entre les balises `<h1> ... </h1>`.

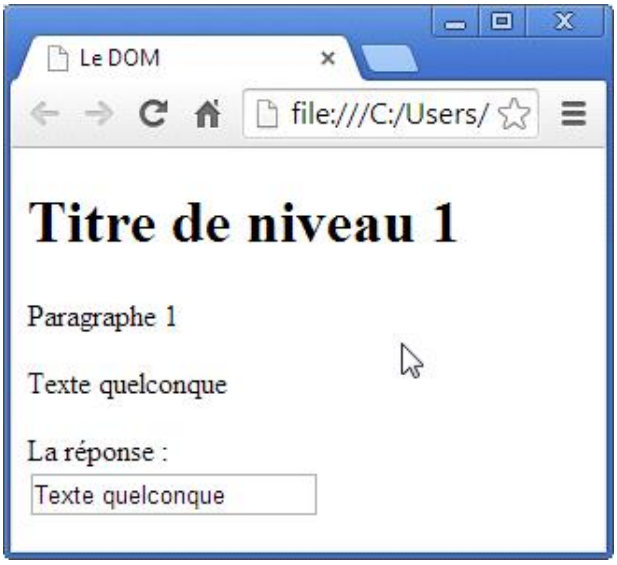
```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Le DOM</title>
<meta charset="UTF-8">
<script>
function valeur() {
var y=document.getElementById("texte");
var x=document.getElementById("titre").firstChild;
y.value=x.nodeValue;
}
</script>
</head>
<body onclick="valeur()">
<h1 id="titre">Titre de niveau 1</h1>
<p>Paragraphe 1</p>
<div>Texte quelconque</div>
<form>
La réponse :<br>
<input id="texte" type="text">
</form>
</body>
</html>
```



L'élément `<h1>` est obtenu par la méthode `getElementById("titre")`. À partir de là, le nœud texte est obtenu par la propriété `firstChild`. Il suffit alors d'afficher la valeur par la propriété `nodeValue`.

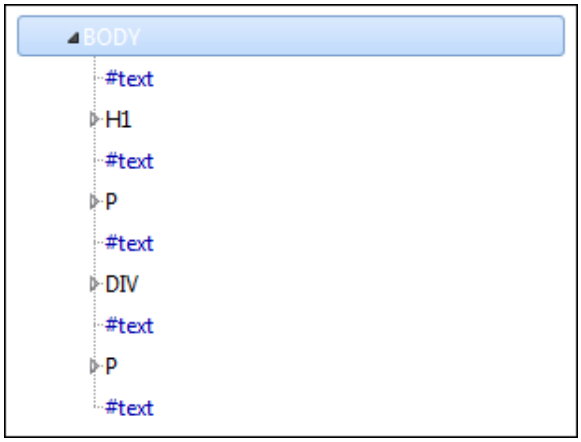
Au clic sur le document, afficher dans la ligne de texte le texte compris entre les balises `<div> ... </div>`.

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Le DOM</title>
<meta charset="UTF-8">
<script>
function valeur() {
var y = document.getElementById("texte");
var x = document.getElementById("titre").parentNode.childNodes[5].
firstChild;
y.value=x.nodeValue;
}
</script>
</head>
<body onclick="valeur()">
<h1 id="titre">Titre de niveau 1</h1>
<p>Paragraphe 1</p>
<div>Texte quelconque</div>
<form>
<p>La réponse :<br>
<input id="texte" type="text"></p>
</form>
</body>
</html>
```



La méthode `getElementById("titre")` accède à la balise `<h1>`. À partir de là, le nœud parent est obtenu par la propriété `parentNodes`, soit la balise `<body>`. On redescend alors vers l'élément enfant `<div>` par `childNodes[5]`. Il faut encore descendre d'un niveau pour atteindre le texte en utilisant `firstChild`. Le texte est affiché par la propriété `nodeValue`.

L'index 5 de `childNodes[5]` peut paraître difficilement compréhensible pour certains. Jetons un coup d'œil à l'arbre du DOM fourni par DOM Inspector.



Le contenu textuel de la balise de paragraphe `<p>` est bien le cinquième élément enfant de la balise parent `<body>`.

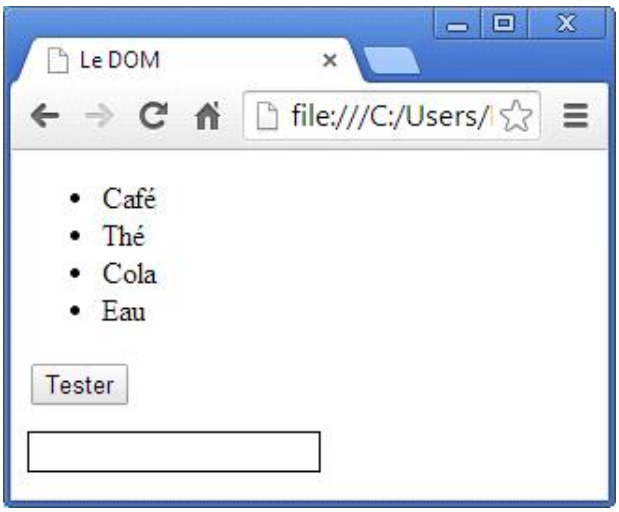
Exemple 2

Soit une liste de quatre items, un bouton et une division :

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Le DOM</title>
<meta charset="UTF-8">
<style>
div { border: 1px solid black;
      width: 150px;
```

```
      padding-left: 5px;
      margin-top: 12px;}
</style>
</head>
<body>
<ul id="liste">
<li id="Item1">Café</li><li id="Item2">Thé</li><li
id="Item3">Cola</li><li id="Item4">Eau</li>
</ul>
<button>Tester</button>
<div id="out"> </div>
</body>
</html>
```

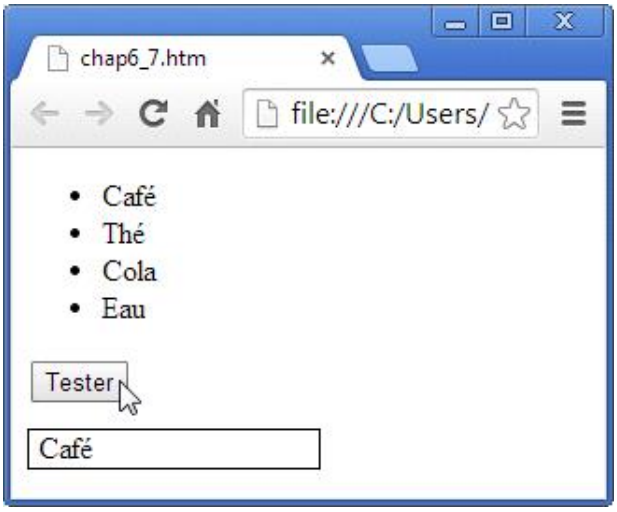
Pour éviter tout problème de compatibilité, il est important que les éléments `<li>` de la liste soient encodés sur la même ligne. Nous évitons ainsi les espaces vides et les passages à la ligne.



Au clic sur le bouton, affichons dans la division le contenu de l'item précédent l'item 2 identifié par `id="item2"`. Pour ce faire, nous utiliserons la méthode `previousSibling()`.

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Le DOM</title>
<meta charset="UTF-8">
<style>
div { border: 1px solid black;
      width: 150px;
      padding-left: 5px;
      margin-top: 12px;}
</style>
</head>
<body>
<ul id="liste">
<li id="Item1">Café</li><li id="Item2">Thé</li><li
id="Item3">Cola</li><li id="Item4">Eau</li>
</ul>
```

```
<button onclick="afficher()">Tester</button>
<div id="out"> </div>
<script>
function afficher(){
var x=document.getElementById("out");
x.innerHTML=document.getElementById("Item2").previousSibling.first
Child.nodeValue;
}
</script>
</body>
</html>
```



Toujours dans le même esprit, affichons, au clic sur le bouton, l'item suivant l'item 2. Nous utiliserons la méthode `nextSibling()`.

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Le DOM</title>
<meta charset="UTF-8">
<style>
div { border: 1px solid black;
      width: 150px;
      padding-left: 5px;
      margin-top: 12px;}
</style>
</head>
<body>
<ul id="liste">
<li id="Item1">Café</li><li id="Item2">Thé</li><li
id="Item3">Cola</li><li id="Item4">Eau</li>
</ul>
<button onclick="afficher()">Tester</button>
<div id="out"> </div>
<script>
function afficher(){
var x=document.getElementById("out");
```

```
x.innerHTML=document.getElementById("Item2")
.nextSibling.firstChild.nodeValue;
}
</script>
</body>
</html>
```