
 Cette page a été traduite à partir de l'anglais par la communauté. Vous pouvez également contribuer en rejoignant la communauté francophone sur [MDN Web Docs](#).

Explorer un tableau HTML avec des interfaces DOM et JavaScript

Introduction

Cet article propose une vue d'ensemble de certaines méthodes DOM Level 1 fondamentales et la façon de les utiliser depuis JavaScript. Vous y apprendrez à créer, accéder, contrôler et supprimer dynamiquement des éléments HTML. Les méthodes DOM décrites ne sont pas spécifiques au HTML et s'appliquent également au XML. Les exemples fonctionneront dans tous les navigateurs offrant le support complet du DOM niveau 1, ce qui est le cas de tous les navigateurs basés sur Mozilla comme Firefox ou Netscape. Les morceaux de code de ce document fonctionneront également dans Internet Explorer 5.

 **Note :** Les méthodes décrites ici font partie de la spécification Document Object Model level 1 (Core). DOM level 1 comprend des méthodes destinées à l'accès et à la manipulation des documents (DOM 1 core) ainsi que des méthodes spécifiques aux documents HTML (DOM 1 HTML).

Création d'un tableau HTML dynamiquement

Contenu HTML

HTML

Play 



```
<input type="button" value="Generate a table." onclick="generate_table()" />
```

Contenu JavaScript

JS

Play 



```
function generate_table() {  
    // get the reference for the body  
    var body = document.getElementsByTagName("body")[0];  
  
    // creates a <table> element and a <tbody> element  
    var tbl = document.createElement("table");  
    var tblBody = document.createElement("tbody");  
  
    // creating all cells  
    for (var i = 0; i < 2; i++) {  
        // creates a table row  
        var row = document.createElement("tr");  
  
        for (var j = 0; j < 2; j++) {  
            // Create a <td> element and a text node, make the text  
            // node the contents of the <td>, and put the <td> at  
            // the end of the table row  
            var cell = document.createElement("td");  
            var cellText = document.createTextNode(  
                "cell in row " + i + ", column " + j,  
            );  
            cell.appendChild(cellText);  
            row.appendChild(cell);  
        }  
  
        // add the row to the end of the table body  
        tblBody.appendChild(row);  
    }  
}
```

```
// put the <tbody> in the <table>
tbl.appendChild(tblBody);
// appends <table> into <body>
body.appendChild(tbl);
// sets the border attribute of tbl to 2;
tbl.setAttribute("border", "2");
}
```

Play 

Generate a table. ...

| | |
|----------------------------------|----------------------------------|
| la cellule est ligne 0 colonne 0 | la cellule est ligne 0 colonne 1 |
| la cellule est ligne 1 colonne 0 | la cellule est ligne 1 colonne 1 |

...

Remarquez l'ordre dans lequel les éléments et le nœud texte sont créés :

1. On crée d'abord l'élément `<table>`.
2. Ensuite, l'élément `<tbody>` qui est un enfant de l'élément `<table>`.
3. Puis, grâce à une boucle, on crée les éléments `<tr>`, qui sont des enfants de l'élément `<tbody>`.
4. Pour chaque élément `<tr>`, on emploie une boucle pour créer les éléments enfants `<td>`.
5. Enfin pour chaque élément `<td>`, on crée le nœud texte contenant le texte de la cellule du tableau.

Après avoir créé les éléments `<table>`, `<tbody>`, `<tr>`, `<td>` et le nœud texte, on ajoute chaque objet à son parent dans l'ordre inverse :

1. On attache d'abord chaque nœud texte à son élément parent `<td>` en

utilisant

JS

Play 



```
cell.appendChild(texte);
```

2. Ensuite, on lie chaque élément `<td>` à son élément `<tr>` parent avec

JS

Play 



```
row.appendChild(cell);
```

3. Puis chaque `<tr>` à son parent `<tbody>` avec

JS

Play 



```
tbody.appendChild(row);
```

4. Puis l'élément `<tbody>` est attaché à son élément parent `<table>` grâce à

JS

Play 



```
table.appendChild(tbody);
```

5. Enfin, `<table>` est rattaché à `<body>` avec

JS

Play 



```
body.appendChild(table);
```

Souvenez-vous de cette technique, vous l'utiliserez souvent en programmant pour le DOM W3C. On crée d'abord les éléments du haut vers le bas, puis on attache les enfants aux parents dans l'ordre inverse.

Voici l'HTML généré par ce code JavaScript :

HTML

Play 



...

```
<table border="2">
```

```
<tr>
```

```
<td>la cellule est ligne 0 colonne 0</td>
```

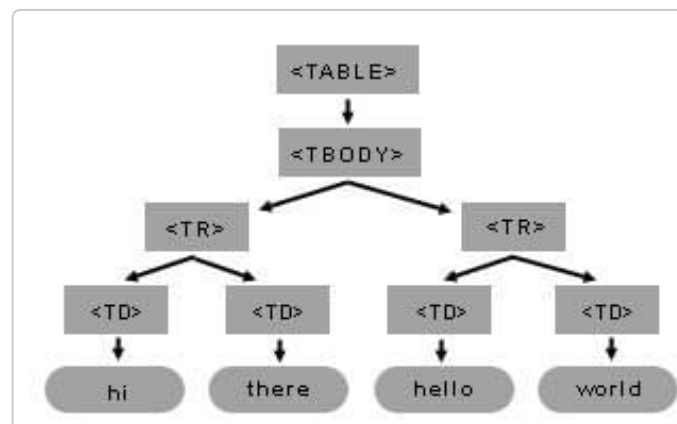
```
<td>la cellule est ligne 0 colonne 1</td>
```

```

</tr>
<tr>
  <td>la cellule est ligne 1 colonne 0</td>
  <td>la cellule est ligne 1 colonne 1</td>
</tr>
</table>
...

```

Voici l'arborescence objet DOM créée par le code, pour l'élément TABLE et ses enfants :



Vous pouvez construire ce tableau ainsi que ses éléments enfants internes en utilisant juste quelques méthodes DOM. Conservez à l'esprit le modèle en arbre des structures que vous comptez créer, cela rendra plus facile l'écriture du code nécessaire. Dans l'arbre `<table>` de la figure 1, l'élément `<table>` a un enfant, l'élément `<tbody>`, qui lui-même a deux enfants `<tr>`, qui à leur tour ont chacun un enfant `<td>`. Enfin, chacun de ces éléments `<td>` a un enfant, un nœud texte.

Définition de la couleur d'arrière-plan d'un paragraphe

`getElementsByTagName` est à la fois une méthode de l'interface `Document` et de l'interface `Element`. Lorsqu'il est appelé, il renvoie un tableau avec tous les descendants de l'élément correspondant au nom de l'étiquette. Le premier élément de la liste se trouve à la position `[0]` dans le tableau.

Contenu HTML

HTML Play

```
<body>
  <input
    type="button"
    value="Set paragraph background color"
    onclick="set_background()" />
  <p>hi</p>
  <p>hello</p>
</body>
```

Contenu JavaScript

JS Play

```
function set_background() {
  // récupère une liste de tous les éléments body (il n'y en aura qu'un),
  // et sélectionne le premier (indice 0) de ces éléments
  myBody = document.getElementsByTagName("body")[0];

  // à présent, trouve tous les éléments p enfants de cet élément body
  myBodyElements = myBody.getElementsByTagName("p");

  // récupère le second élément de cette liste d'éléments p
  myP = myBodyElements[1];
  myP.style.background = "rgb(255,0,0)";
}
```

Play

Set paragraph background color


hi

hello

```
myTextNode = document.createTextNode("world");
```


Dans cet exemple, on assigne à la variable `myP` l'objet DOM du second élément `p` du corps (body).

1. On récupère d'abord une liste de tous les éléments body avec


```
JS Play   
myBody = document.getElementsByTagName("body")[0];
```

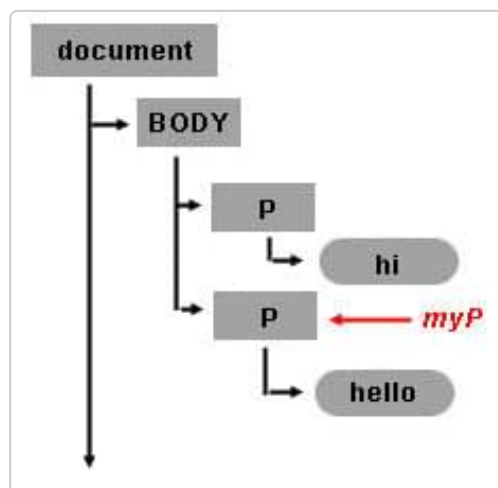
Puisqu'il n'existe qu'un seul élément body dans un document HTML valide, cette liste ne comporte qu'un élément, que l'on récupère en sélectionnant le premier élément de la liste grâce à `[0]`.

2. Ensuite, on récupère tous les éléments `p` qui sont des enfants de body en utilisant

```
JS Play   
myBodyElements = myBody.getElementsByTagName("p");
```

3. Pour finir on prend le deuxième élément de la liste des éléments `p` avec

```
JS Play   
myP = myBodyElements[1];
```



Une fois que vous avez l'objet DOM pour un élément HTML, vous pouvez modifier ses propriétés. Si par exemple vous voulez définir la propriété couleur d'arrière-plan du style, ajoutez simplement :

JS

Play 

```
myP.style.background = "rgb(255,0,0)";  
// ajoute une propriété de style inline
```

Création de nœuds texte avec `document.createTextNode("...")`

Employez l'objet `document` pour appeler la méthode `createTextNode` et créer un nœud texte. Il suffit de lui communiquer le contenu texte, et la valeur renvoyée est un objet représentant le nœud texte.

HTML

Play 

```
myTextNode = document.createTextNode("world");
```

Ce morceau de code crée un nœud de type `TEXT_NODE` qui contient la donnée texte `"world"`, et `monNoeudTexte` est la référence de l'objet nœud créé. Pour afficher ce texte sur votre page HTML, vous devez ensuite définir ce nœud texte comme l'enfant d'un autre élément nœud.

Insertion d'éléments avec `appendChild(...)`

En invoquant `myP.appendChild('node_element')`, vous définissez `node_element` comme un nouvel enfant du second élément `p` (`myP` a été défini plus haut comme étant le second élément `p`).

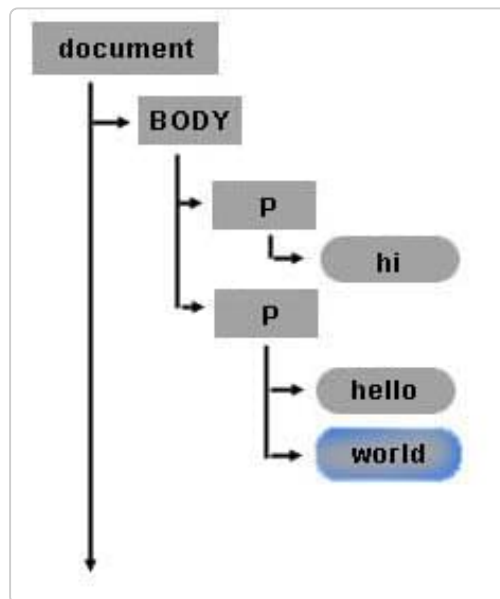
JS

Play 

```
myP.appendChild(noeudTexte);
```

En exécutant cet exemple, vous pouvez remarquer que les mots « hello » et « world » ne sont pas séparés : `helloworld`. Quand vous parcourez la page HTML les deux nœuds semblent donc n'en former qu'un seul, rappelez-vous cependant qu'ils sont bien distincts dans le modèle de document. Le second nœud est de

type `TEXT_NODE`, et est le second enfant de la seconde balise `<p>`. Le schéma suivant situe ce nouvel objet dans l'arborescence du document :

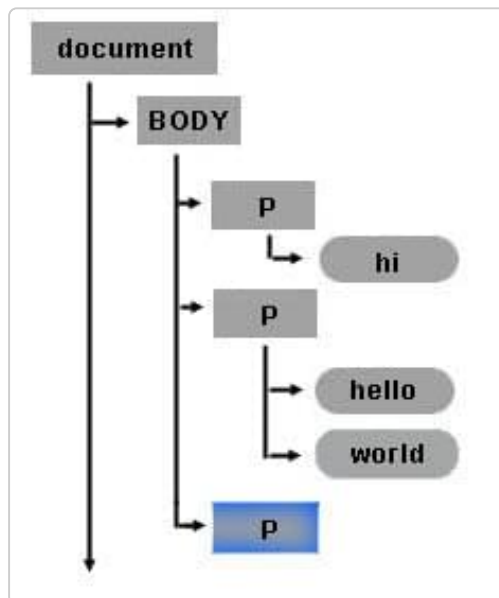


Note : L'utilisation de `createTextNode` et de `appendChild` permet aisément d'ajouter un espace entre ces deux mots. Notez cependant que la méthode `appendChild` ajoute le nouvel enfant à la suite de ceux déjà présents, à la manière de « world » placé après « hello ». Pour ajouter un nœud texte entre « hello » et « world » (par exemple un espace), utilisez `insertBefore` à la place de `appendChild`.

Création de nouveaux éléments avec l'objet `document` et la méthode `createElement(...)`

Vous pouvez créer de nouveaux éléments, dont des éléments HTML, avec `createElement`. Pour créer un élément `<p>` enfant de l'élément `<body>`, vous pouvez vous servir de `body` défini dans l'exemple précédent et lui greffer un nouvel élément nœud. Pour ce faire, invoquez `document.createElement("nombalise")`. Voici un exemple :

```
nouveauNoeudBALISEP = document.createElement("p");  
body.appendChild(nouveauNoeudBALISEP);
```



Suppression de nœuds avec la méthode `removeChild(...)`

Tous les nœuds peuvent être supprimés. La ligne ci-dessous supprime de `myP` (deuxième élément `<p>`) le nœud texte contenant le mot « world » :

JS

[Play](#)



```
myP.removeChild(noeudTexte);
```

Vous pouvez ensuite ajouter `monNoeudTexte` (contenant "world") dans l'élément `<p>` récemment créé :

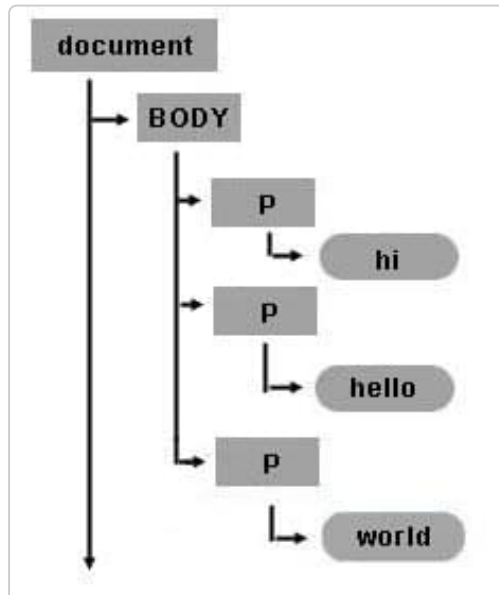
JS

[Play](#)



```
nouveauNoeudBALISEP.appendChild(noeudTexte);
```

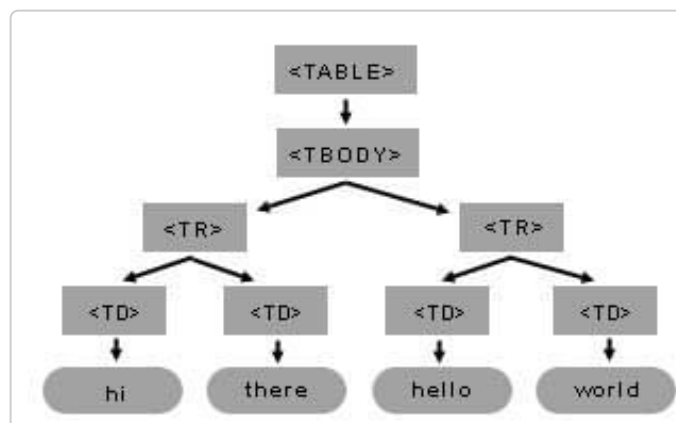
L'arborescence des objets se présente désormais comme ceci :



Création dynamique d'un tableau (retour à Sample1.html)

Jusqu'à la fin de cet article, nous travaillons de nouveau sur Exemple1.html. Le schéma qui suit vous rappelle la structure de l'arbre des objets pour le tableau créé dans l'exemple.

Rappel de la structure arborescente d'un tableau HTML



Création et insertion des éléments dans l'arborescence

On peut décomposer la création du tableau de Exemple1.html en trois étapes :

- Récupérer l'objet body (c'est le premier élément de l'objet document).
- Créer tous les éléments.
- Greffer chaque enfant sur son parent en respectant la structure du tableau (cf. le schéma ci-dessus).

Le code source qui suit est un exemple commenté qui crée le tableau de Exemple1.

Note : Il y a une ligne de code supplémentaire à la fin de la fonction `start()`, qui définit la propriété bordure du tableau en employant la méthode `setAttribute`. `setAttribute` utilise deux arguments : le nom de l'attribut et sa valeur, et permet de définir n'importe quelle propriété de n'importe quel élément.

HTML



```
<head>
<title>Code de démonstration – Explorer un tableau HTML avec des interfaces
DOM et JavaScript</title>
<script>
    function start() {
        // récupère une référence vers l'élément body
        var body = document.getElementsByTagName("body")[0];

        // création des éléments <table> et <tbody>
        table      = document.createElement("table");
        tablebody  = document.createElement("tbody");

        // création des cellules
        for(var j = 0; j < 2; j++) {
            // création d'un élément <tr>
            row = document.createElement("tr");

            for(var i = 0; i < 2; i++) {
                // création d'un élément <td>
```

```

        cell = document.createElement("td");
        // création d'un nœud texte
        texte = document.createTextNode("la cellule est ligne " + j +
", colonne " + i);
        // ajoute le nœud texte créé à la cellule <td>
        cell.appendChild(texte);
        // ajoute la cellule <td> à la ligne <tr>
        row.appendChild(cell);
    }
    // ajoute la ligne <tr> à l'élément <tbody>
    tablebody.appendChild(row);
}

// ajoute <tbody> à l'élément <table>
table.appendChild(tablebody);
// ajoute <table> à l'élément <body>
body.appendChild(table);
// définit l'attribut border de table à 2;
table.setAttribute("border", "2");
}
</script>
</head>
<body onload="start()">
</body>
</html>

```

Manipulation du tableau avec DOM et CSS

Récupérer un nœud texte dans le tableau

Cet exemple présente deux nouveaux attributs DOM. D'abord, l'attribut `childNodes` qui est utilisé pour récupérer la liste des nœuds enfants de `cel`. A la différence de `getElementsByTagName`, la liste renvoyée par `childNodes` comporte tous les enfants sans considération de type. Une fois la liste obtenue, la notation `[x]` est employée pour sélectionner l'élément enfant désiré. Dans cet exemple, le nœud texte de la seconde cellule de la seconde ligne du tableau est enregistré dans `celtext`. Ensuite, un nouveau nœud texte contenant les données de

`celtext` est greffé en tant qu'enfant sur l'élément `<body>`.

Note : Si l'objet est un nœud texte, vous pouvez récupérer le texte qu'il contient en employant l'attribut `data`.

JS



```
mybody = document.getElementsByTagName("body")[0];
mytable = mybody.getElementsByTagName("table")[0];
mytablebody = mytable.getElementsByTagName("tbody")[0];
myrow = mytablebody.getElementsByTagName("tr")[1];
mycel = myrow.getElementsByTagName("td")[1];

// premier élément du noeud liste des enfants de mycel
myceltext = mycel.childNodes[0];

// contenu de currenttext est le contenu des données de myceltext
currenttext = document.createTextNode(mycltext.data);
mybody.appendChild(currenttext);
```

Récupérer la valeur d'un attribut

A la fin d'Exemple1, l'appel à `setAttribute` sur l'objet `table` définit la propriété `border` du tableau. Si vous désirez simplement récupérer la valeur de cet attribut, vous pouvez employer la méthode `getAttribute` :

HTML



```
mytable.getAttribute("border");
```

Cacher une colonne en changeant les propriétés de style

Une fois que vous avez l'objet dans une variable JavaScript, vous pouvez définir les propriétés directement. Le code qui suit est une version modifiée de Exemple1.html où les cellules de la seconde colonne sont cachées, et le fond de celles de la première colonne est rouge. Remarquez que la propriété de style `y` est

définie directement.

HTML 

```
<html>
  <body onload="start()"></body>
  <script>
    function start() {
      var body = document.getElementsByTagName("body")[0];
      table = document.createElement("table");
      tablebody = document.createElement("tbody");


      for (var j = 0; j < 2; j++) {
        row = document.createElement("tr");
        for (var i = 0; i < 2; i++) {
          cell = document.createElement("td");
          text = document.createTextNode("la cellule est :" + i + j);
          cell.appendChild(text);
          row.appendChild(cell);
          // change la couleur de fond de la cellule
          // si la colonne est 0. Si la colonne est 1, cache la cellule
          if (i == 0) {
            cell.style.background = "rgb(255,0,0)";
          } else {
            cell.style.display = "none";
          }
        }
        tablebody.appendChild(row);
      }
      table.appendChild(tablebody);
      body.appendChild(table);
    }
  </script>
</html>
```

Original Document Information

Author(s)

Marcio Galli

Migrated from

<http://web.archive.org/web/20000815054125/http://mozilla.org/docs/dom/tech-note/tn-dom-table/> 

Interwik

This page was last modified on 4 août 2023 by [MDN contributors](#).