

JavaScript - Arbre DOM -

Groupe des étudiants : CIR1

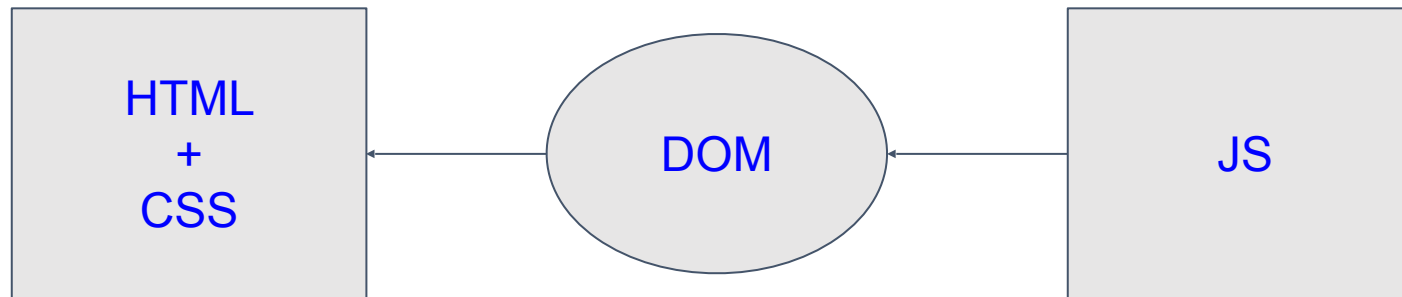


Définition et exemples



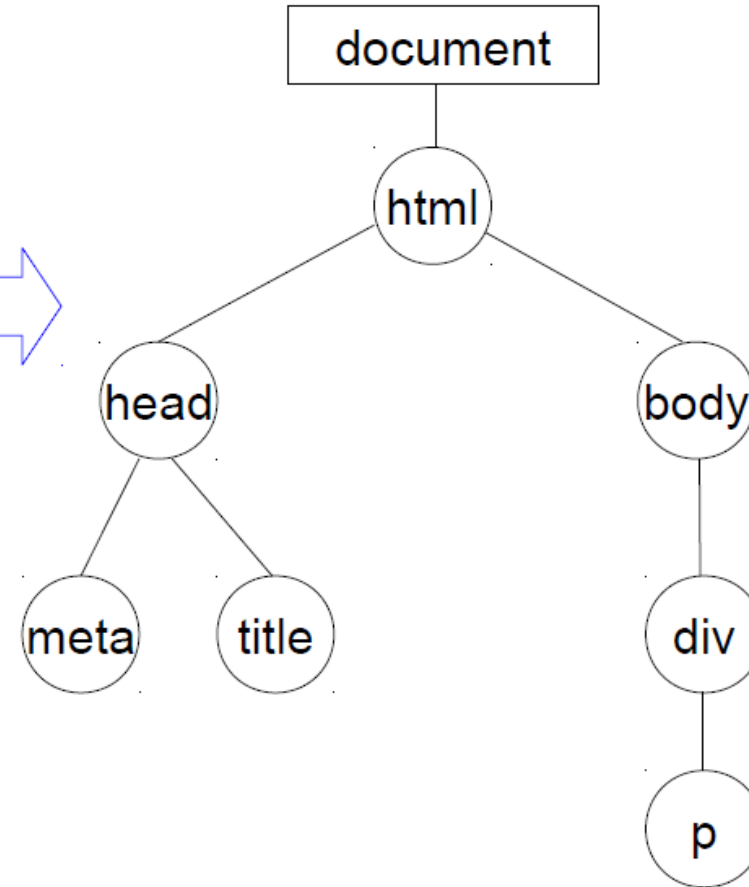
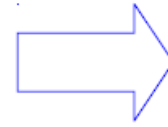
Définition

- En mémoire, une page web est représentée sous la forme d'un arbre, appelé arbre **DOM** (Document Object Model)
- Une fois l'arbre construit en mémoire, on peut le **consulter** et le **modifier** dynamiquement avec JavaScript
- La racine de l'arbre DOM est toujours stockée dans la variable globale **document**



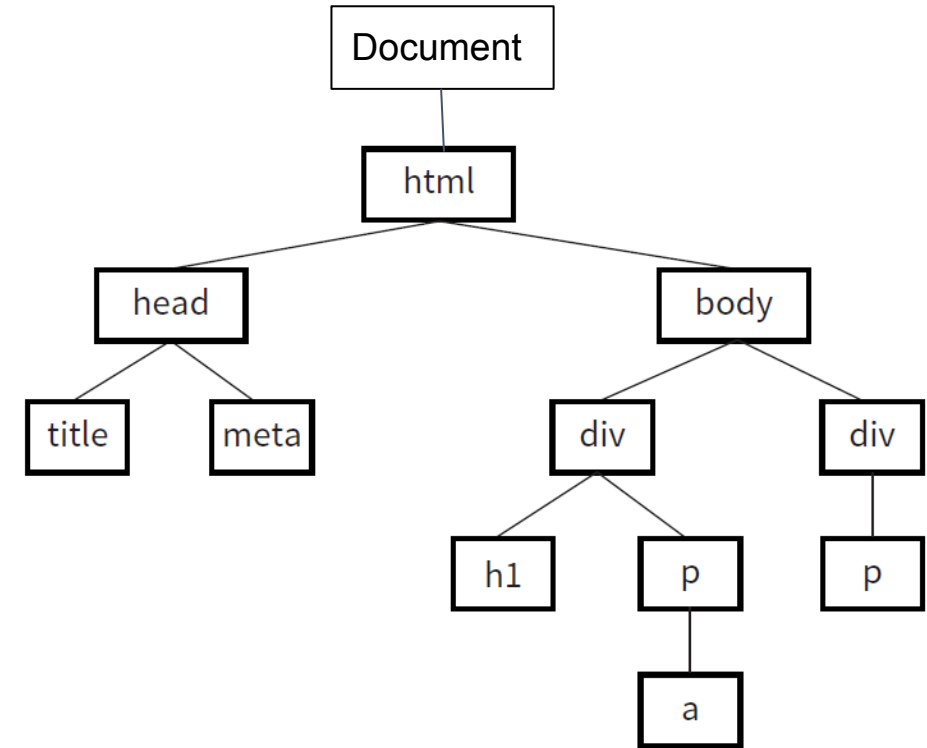
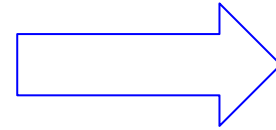
Exemple 1

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <title>A web page</title>
  </head>
  <body>
    <div>
      <p>Paragraphe</p>
    </div>
  </body>
</html>
```



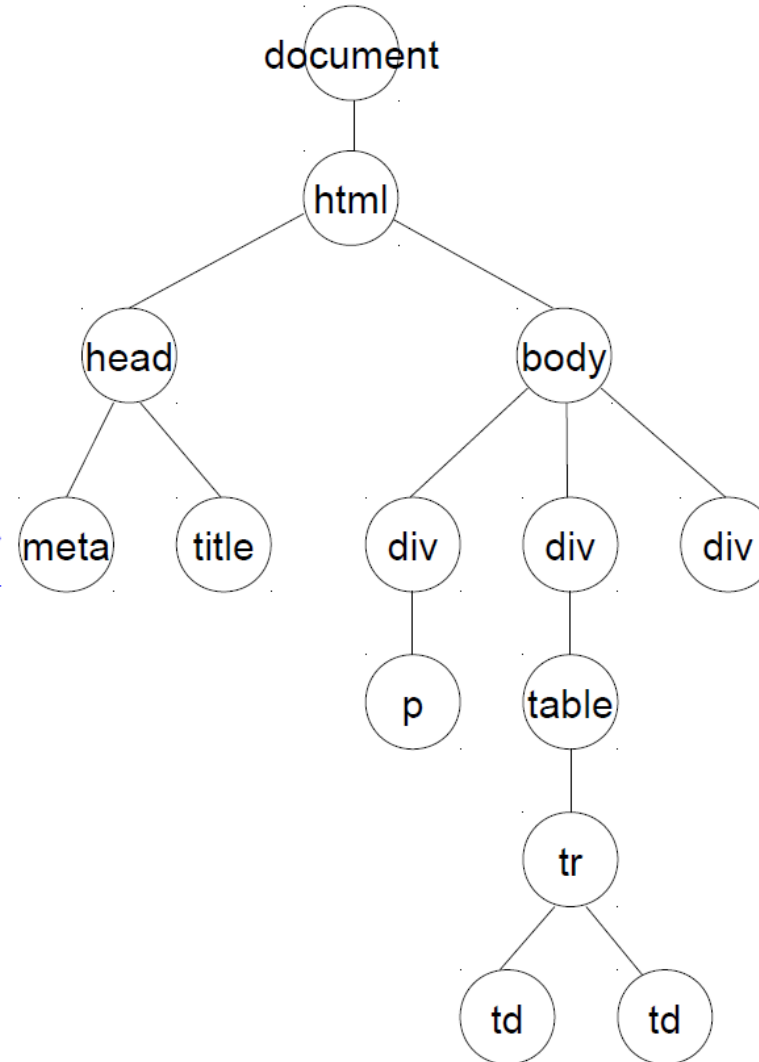
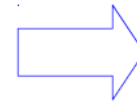
Exemple 2

```
<!DOCTYPE html>
<html>
<head>
  <title>Document title</title>
  <meta charset="utf-8">
</head>
<body>
  <div>
    <h1>Heading</h1>
    <p>Paragraph text with a <a href="foo.html">link</a> here.</p>
  </div>
  <div>
    <p>More text here.</p>
  </div>
</body>
</html>
```



Exemple 3

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <title>A web page</title>
  </head>
  <body>
    <div>
      <p>Paragraphe</p>
    </div>
    <div>
      <table>
        <tr>
          <td>Colonne 1</td>
          <td>Colonne 2</td>
        </tr>
      </table>
    </div>
    <div>Div</div>
  </body>
</html>
```



Accès aux éléments du DOM

Raccourcis pratiques

- La variable globale `document` est de type `Document`. Elle contient des attributs qui pointent **directement** sur certains éléments de la page :

<code><u>document.body</u></code>	élément correspondant à la balise <code><body></code> de la page
<code>document.head</code>	élément correspondant à la balise <code><head></code> de la page
<code>document.images</code>	éléments correspondant aux balises <code></code> de la page
<code><u>document.links</u></code>	éléments correspondant aux balises <code><a></code> de la page
<code><u>document.title</u></code>	éléments correspondant aux balises <code><title></code> de la page

Recherche par propriété

- La variable globale `document` est de type `Document`. Elle contient des fonctions permettant de rechercher des éléments par **propriété** :

<code>getElementById(id)</code>	renvoie l'élément qui possède l'identifiant spécifié
<code>getElementsByTagName(tag)</code>	renvoie les éléments ayant le nom de balise spécifié
<code>getElementsByClassName(class)</code>	renvoie les éléments ayant le nom de classe CSS spécifié

Recherche par sélection CSS

- La variable globale `document` est de type `Document`. Elle contient des fonctions permettant de rechercher des éléments par **sélection CSS (id, classe, nom, ...)** :

<code>querySelector(selector)</code>	renvoie le premier élément trouvé correspondant au sélecteur CSS spécifié
<code>querySelectorAll(selector)</code>	renvoie tous les éléments correspondant au sélecteur CSS spécifié

Exemple 1

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <title>A web page</title>
  </head>
  <body>
    <div id="i1" class="c2">
      <p>Paragraphe</p>
    </div>
    <div class="c1">
      <table>
        <tr class="c2">
          <td>Colonne 1</td>
          <td>Colonne 2</td>
        </tr>
      </table>
    </div>
    <div id="i2" class="c2">Div</div>
  </body>
</html>
```

```
document.querySelector("#i1");
```

ou

```
document.getElementById("i1");
```

Exemple 2

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <title>A web page</title>
  </head>
  <body>
    <div id="i1" class="c2">
      <p>Paragraphe</p>
    </div>
    <div class="c1">
      <table>
        <tr class="c2">
          <td>Colonne 1</td>
          <td>Colonne 2</td>
        </tr>
      </table>
    </div>
    <div id="i2" class="c2">Div</div>
  </body>
</html>
```

```
document.querySelectorAll(".c2");
```

ou

```
document.getElementsByClassName("c2");
```

Exemple 3

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <title>A web page</title>
  </head>
  <body>
    <div id="i1" class="c2">
      <p>Paragraphe</p>
    </div>
    <div class="c1">
      <table>
        <tr class="c2">
          <td>Colonne 1</td>
          <td>Colonne 2</td>
        </tr>
      </table>
    </div>
    <div id="i2" class="c2">Div</div>
  </body>
</html>
```

```
document.querySelectorAll("div");
```

ou

```
document.getElementsByTagName("div");
```

Exemple 4

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <title>A web page</title>
  </head>
  <body>
    <div id="i1" class="c2">
      <p>Paragraphe</p>
    </div>
    <div class="c1">
      <table>
        <tr class="c2">
          <td>Colonne 1</td>
          <td>Colonne 2</td>
        </tr>
      </table>
    </div>
    <div id="i2" class="c2">Div</div>
  </body>
</html>
```

```
document.querySelectorAll("div.c2");
```

Exemple 5

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <title>A web page</title>
  </head>
  <body>
    <div id="i1" class="c2">
      <p>Paragraphe</p>
    </div>
    <div class="c1">
      <table>
        <tr class="c2">
          <td>Colonne 1</td>
          <td>Colonne 2</td>
        </tr>
      </table>
    </div>
    <div id="i2" class="c2">Div</div>
  </body>
</html>
```

```
document.querySelectorAll("div+div");
```

Affichage du contenu d'un élément

HTML

```
<p id="i1"> hello world </p>
```

JS

```
let e = document.getElementById('i1');  
console.log(e.innerHTML);
```

console

```
hello world
```

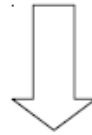
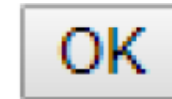

Modification d'éléments via l'arbre DOM

Attributs HTML

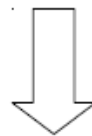
<code>element.hasAttribute(name)</code>	Renvoie <code>true</code> si <code>element</code> possède l'attribut <code>name</code> , <code>false</code> sinon.
<code>element.getAttribute(name)</code>	Renvoie la valeur de l'attribut <code>name</code> de <code>element</code> (<code>" "</code> ou <code>null</code> s'il n'existe pas)
<code>element.setAttribute(name, value)</code>	Affecte <code>value</code> à l'attribut <code>name</code> de <code>element</code> (crée l'attribut s'il n'existait pas)
<code>element.removeAttribute(name)</code>	Supprime l'attribut <code>name</code> de <code>element</code> .

■ Exemple :

```
<button id="ok">OK</button>
```



```
document.querySelector("#ok").setAttribute("disabled", "true");
```



```
<button id="ok" disabled="true">OK</button>
```



Autres propriétés

<code>element.id</code>	Identifiant de <code>element</code> (plutôt destiné à être utilisé lors de la création d'un élément)
<code>element.innerText</code> <code>element.textContent</code>	Contenu textuel de <code>element</code> (quelques différences de comportement, notamment dans la prise en compte du style)
<code>element.className</code>	Nom des classes CSS appliquées à <code>element</code> , séparées par des espaces.
<code>element.style</code>	Objet contenant les propriétés CSS de <code>element</code> . La propriété CSS <code>p</code> est manipulable via <code>element.style.p</code>

Autres propriétés

```
<p id="p42" class="yellow">A paragraph</p>
```

A paragraph



```
document.querySelector("#p42").className = "green";
```



```
<p id="p42" class="green">A paragraph</p>
```

A paragraph

```
.green{  
  background-color: green;  
}
```

Création et suppression d'éléments via l'arbre DOM



Ajout et suppression de noeuds

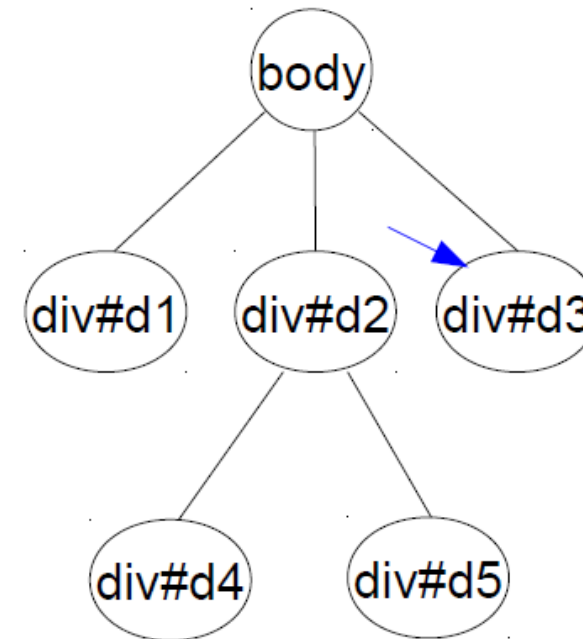
Appel	Effet
<code>node.removeChild(child)</code>	Retire <code>child</code> des enfants de <code>node</code> .
<code>node.appendChild(child)</code>	Ajoute <code>child</code> à la fin des enfants de <code>node</code> .
<code>node.replaceChild(newChild, oldChild)</code>	Remplace <code>oldChild</code> par <code>newChild</code> dans les enfants de <code>node</code> .
<code>node.insertBefore(child, referenceNode)</code>	Insère <code>child</code> dans les enfants de <code>node</code> , juste avant <code>referenceNode</code> .

Les noeuds en argument peuvent être :

- des noeuds existants (dans ce cas ils seront déplacés)
- des noeuds créés dynamiquement

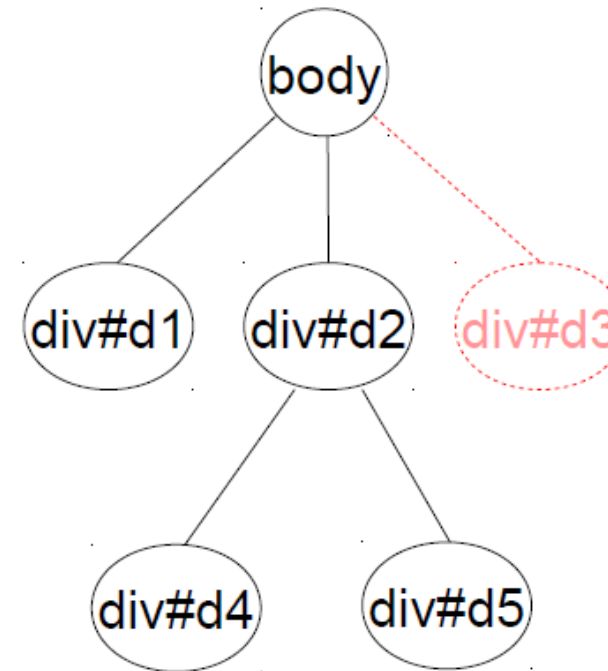
Ajout et suppression de noeuds

```
let d3 = document.querySelector("#d3");
```



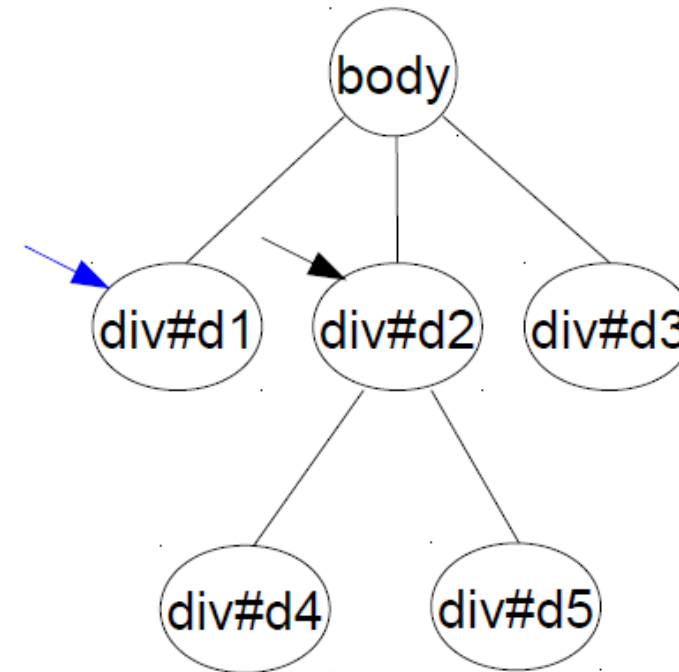
Ajout et suppression de noeuds

```
let d3 = document.querySelector("#d3");  
document.body.removeChild(d3);
```



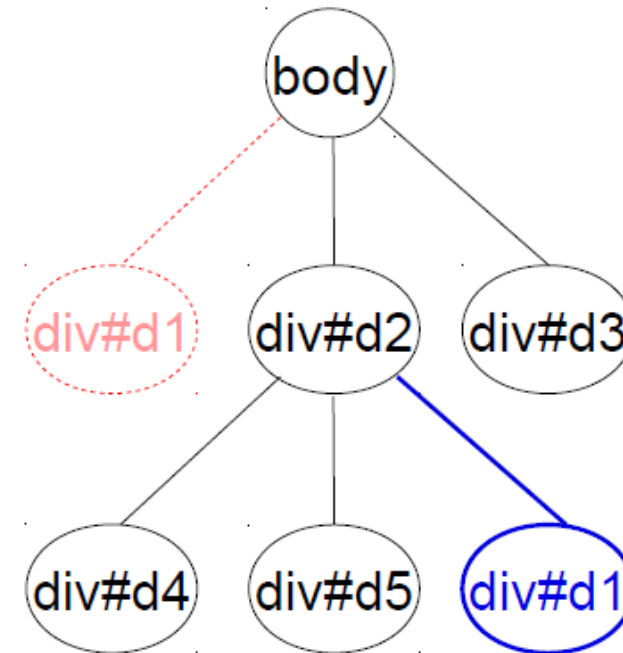
Ajout et suppression de noeuds

```
let d1 = document.querySelector("#d1");  
let d2 = document.querySelector("#d2");
```



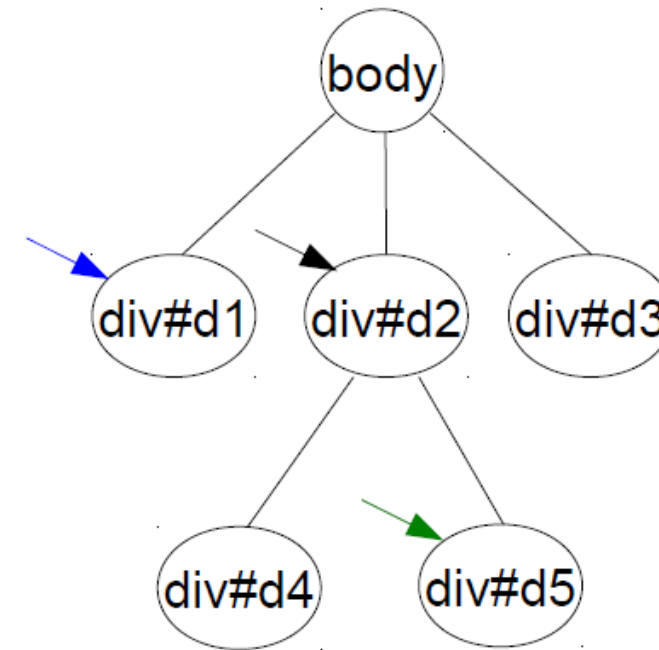
Ajout et suppression de noeuds

```
let d1 = document.querySelector("#d1");  
let d2 = document.querySelector("#d2");  
d2.appendChild(d1);
```



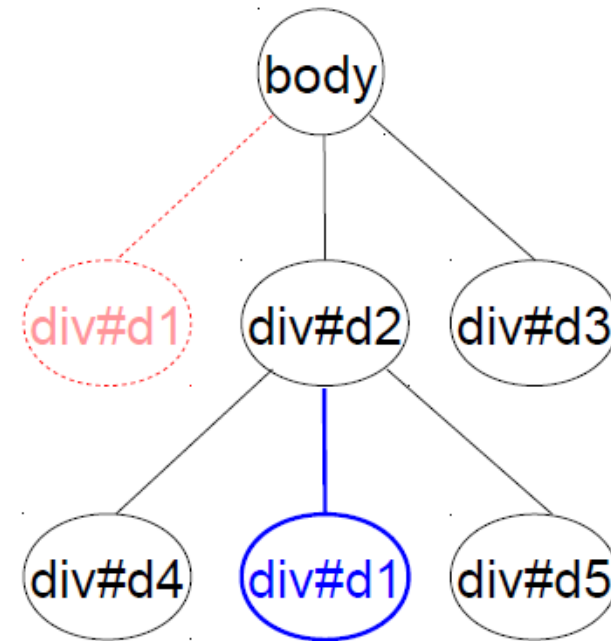
Ajout et suppression de noeuds

```
let d1 = document.querySelector("#d1");  
let d2 = document.querySelector("#d2");  
let d5 = document.querySelector("#d5");
```



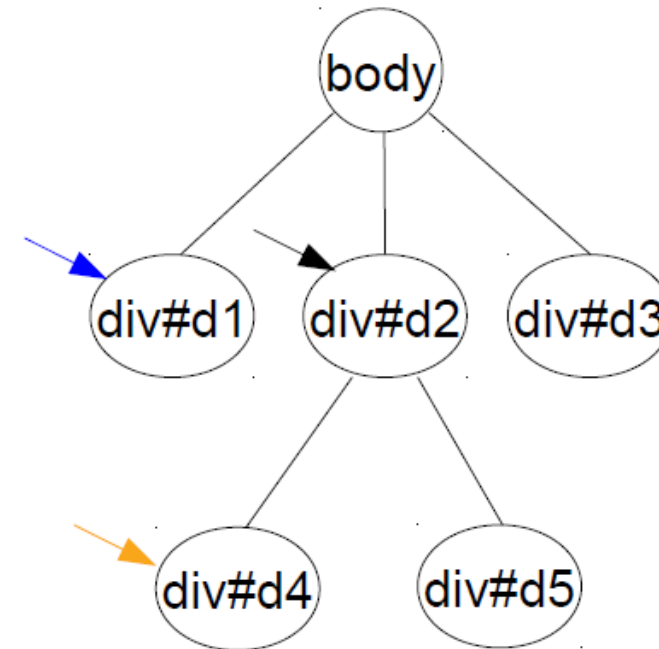
Ajout et suppression de noeuds

```
let d1 = document.querySelector("#d1");  
let d2 = document.querySelector("#d2");  
let d5 = document.querySelector("#d5");  
d2.insertBefore(d1, d5);
```



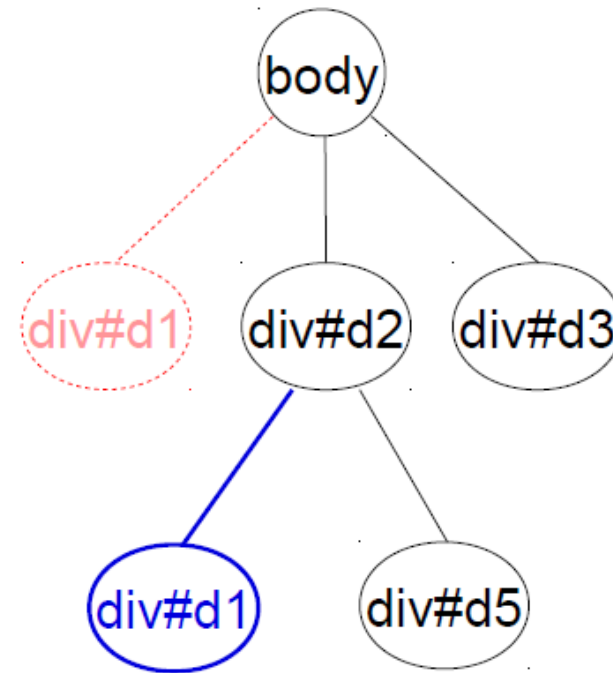
Ajout et suppression de noeuds

```
let d1 = document.querySelector("#d1");  
let d2 = document.querySelector("#d2");  
let d4 = document.querySelector("#d4");
```



Ajout et suppression de noeuds

```
let d1 = document.querySelector("#d1");  
let d2 = document.querySelector("#d2");  
let d4 = document.querySelector("#d4");  
d2.replaceChild(d1, d4);
```



Création dynamique d'élément

- `document.createElement(tagName)` permet de créer un élément HTML de type `tagName`
- Exemple:

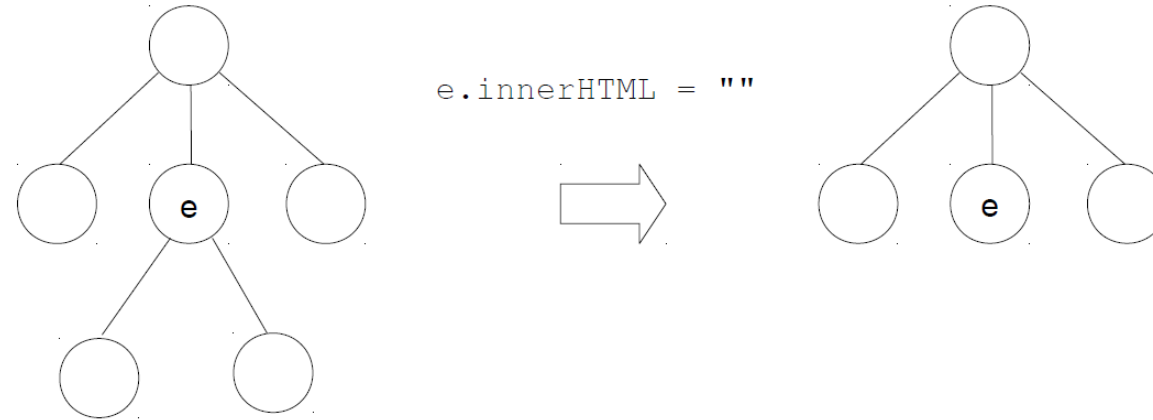
```
let button = document.createElement("button");  
button.id = "uniqueId"; //pour retrouver ce bouton par la suite  
button.innerText = "Dynamically created button";
```

Créera le bouton :

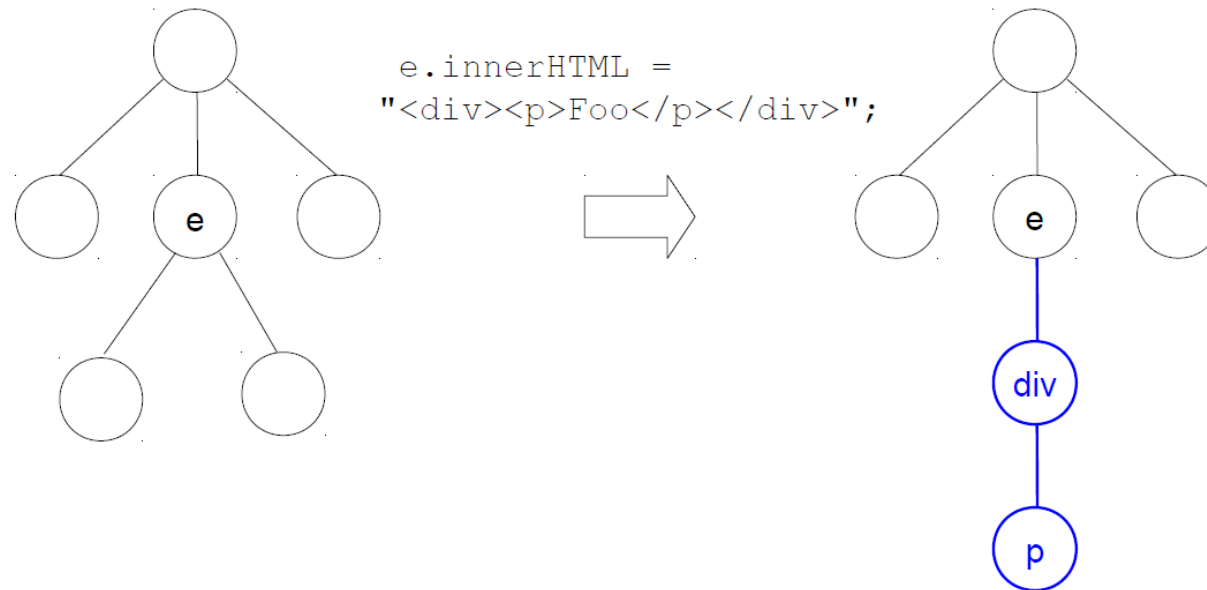
Dynamically created button

innerHTML

■ Suppression

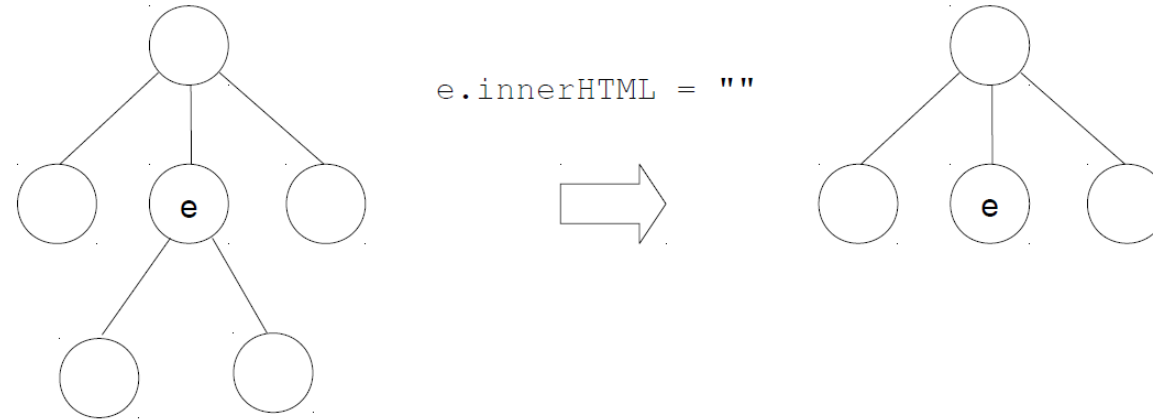


■ Ajout

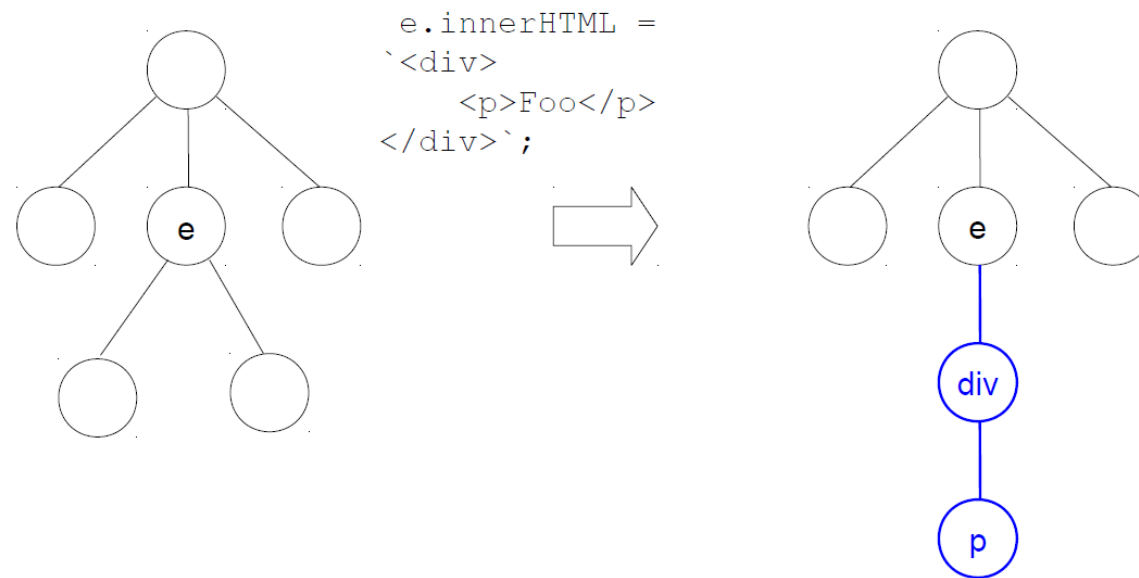


innerHTML

■ Suppression



■ Ajout



Utilisation d'un
gabarit de chaîne