IIT
INSTITUT INTERNATIONAL TECHNOLOGIE
Université Nord Américaine Privée

Programmation web II

Mohamed Karim Abdmouleh
karim.abdelmoula@iit.ens.tn

1ère année Génie Informatique

(Version 0)

Chapitre 2 : DOM

Plan du chapitre

- Qu'est ce que le DOM ?
- Comment accéder aux éléments de l'arbre du document ?
- Comment modifier le document ?
- Comment modifier le style du document ?
- Comment travailler avec les événements ?

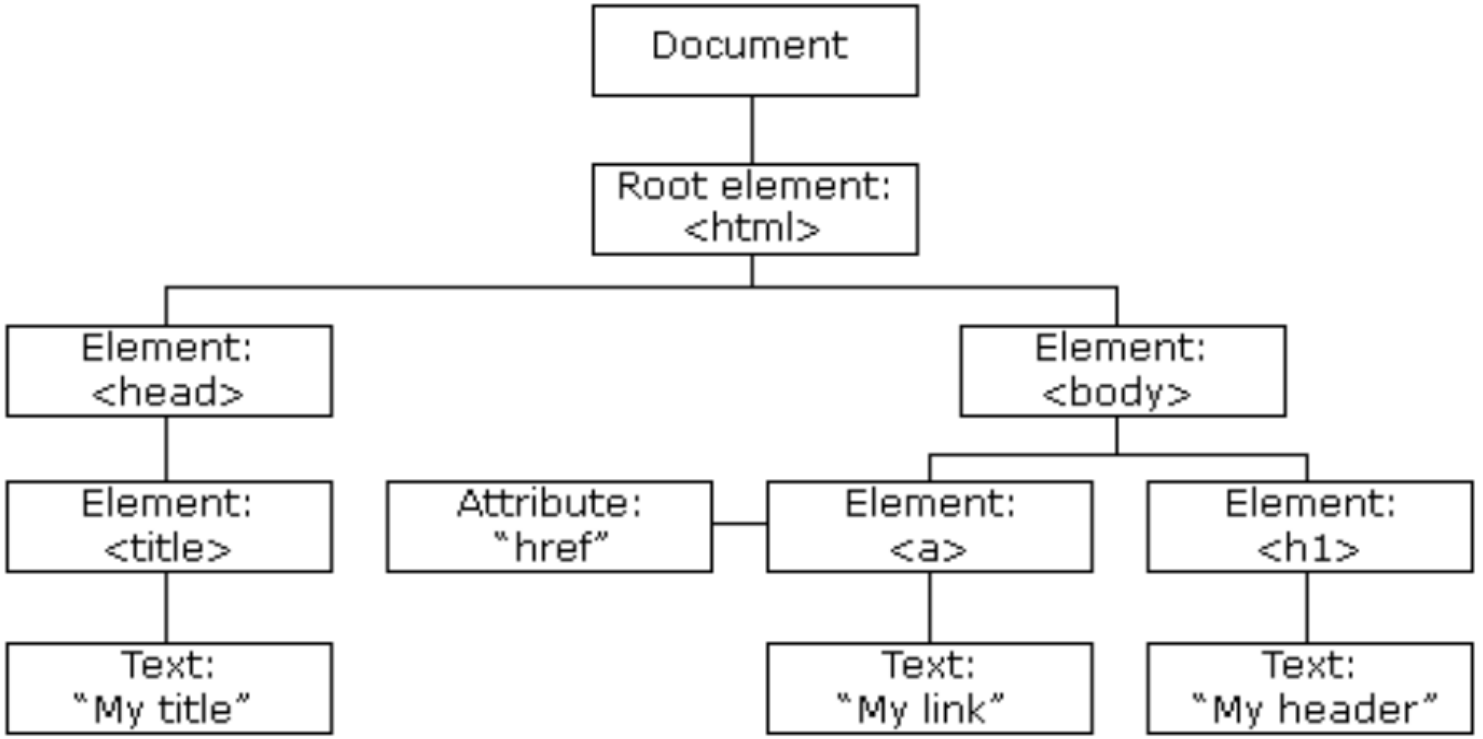
Mohamed Karim Abdmouleh
2025

DOM = Document Object Model

- API (Application Programming Interface) pour la manipulation de HTML / XML
- Le DOM permet :
  - de construire une représentation en **arbre** d'un document
  - d'offrir une **interface commune d'accès** et **modification** du document, quelque soit le langage applicatif
- Avec DOM, on peut
  - Créer des documents
  - Parcourir leur structure
  - Ajouter, effacer, modifier des éléments
  - Ajouter, effacer, modifier leur contenu

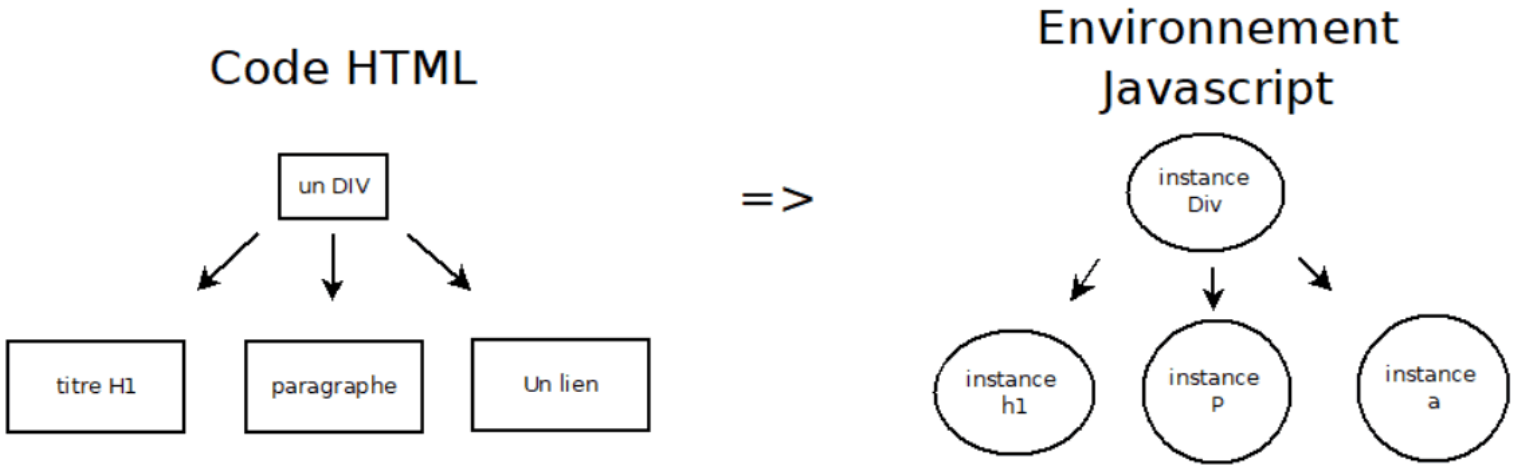
L’arbre DOM

```
<html>
 <head>
 <title>My title</title>
 </head>
 <body>
 <h1>My header</h1>
 <a href=”google.com”>My link</a>
 </body>
</html>
```



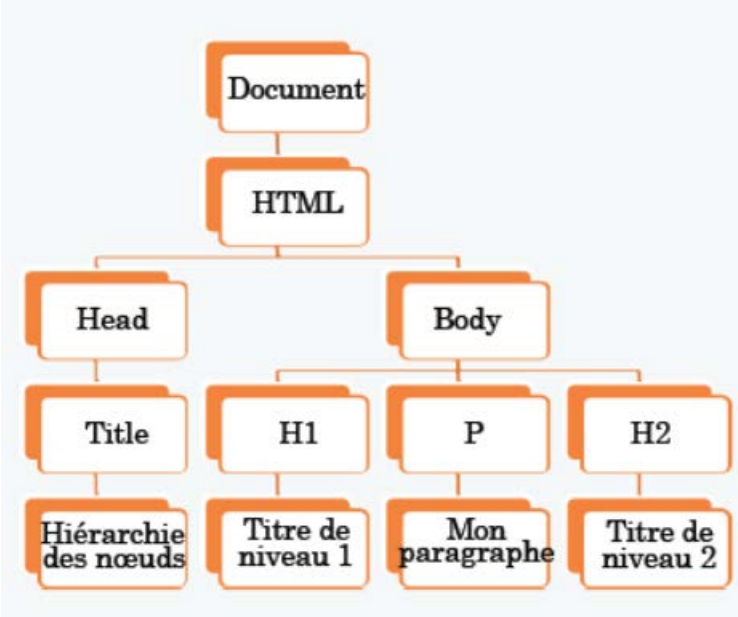
Comment fonctionne l'interface DOM ?

1. Le document HTML est analysé par le navigateur.
2. Pour chaque élément HTML trouvé, le navigateur instancie un objet JavaScript et le place dans un arbre. Il constitue ainsi un arbre d'instances d'objets JavaScript à partir du contenu HTML.
3. L'arbre est mis à disposition du développeur à partir d'une instance d'un objet spécifique qui s'appelle **document**.



Hiérarchie des noeuds

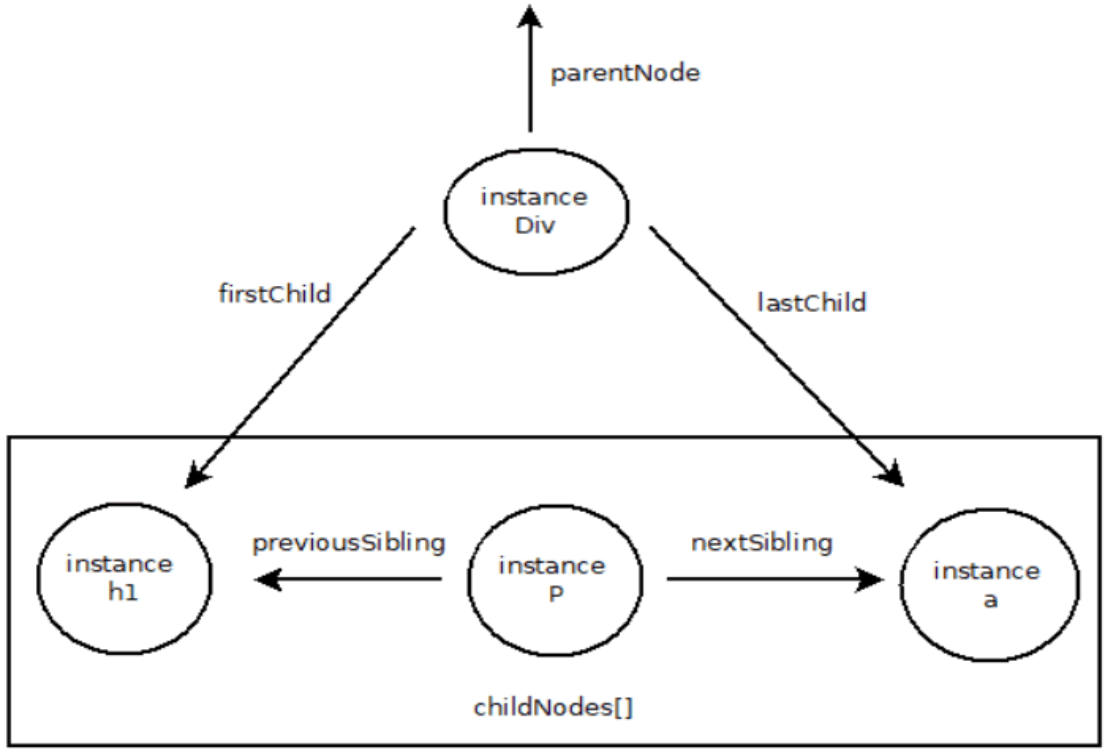
- Tous les éléments d'un document HTML sont des noeuds :
    - Tout le document est un "**document node**"
    - Tout élément html est un "**element node**" e.g. *html, head, body, title, …*
    - Le texte dans un élément HTML un "**text node**" e.g. *Hiérarchie des nœuds, …*
    - Les attributs d'un élément HTML sont des "**attribute nodes**"



Mohamed Karim Abdmouleh
2025

La structure d'un document HTML en arbre

- Le premier noeud de l'arbre est appelé la **racine**
- Tout noeud, sauf la racine, a un **seul** noeud père
- Un noeud peut avoir **plusieurs fils**
- Une feuille est un **noeud sans fils**
- Les frères sont des noeuds ayant le **même** père



Les **propriétés** des noeuds

- Tout élément HTML DOM possède des propriétés auquel on peut accéder et/ou modifier avec **JavaScript**, exemples :
- x.innerHTML – la valeur du texte dans le noeud x
- x.nodeName – le nom du noeud x
- x.nodeValue – la valeur du noeud x
- x.parentNode – le noeud père du noeud x
- x.childNodes – les noeuds fils du noeud x
- x.attributes – les attributs du noeud x

Les propriétés des noeuds

- La propriété **innerHTML**: accès au contenu des éléments HTML : c-à-d à ce qui se situe entre les deux balises ouvrante et fermante d'un élément
- La propriété **textContent**: pour récupérer *que le contenu textuel* présent à l'intérieur d'un élément.
- **innerHTML** et **textContent** sont des propriétés de **l'objet Element** et non pas Document.

Mohamed Karim Abdmouleh
2025

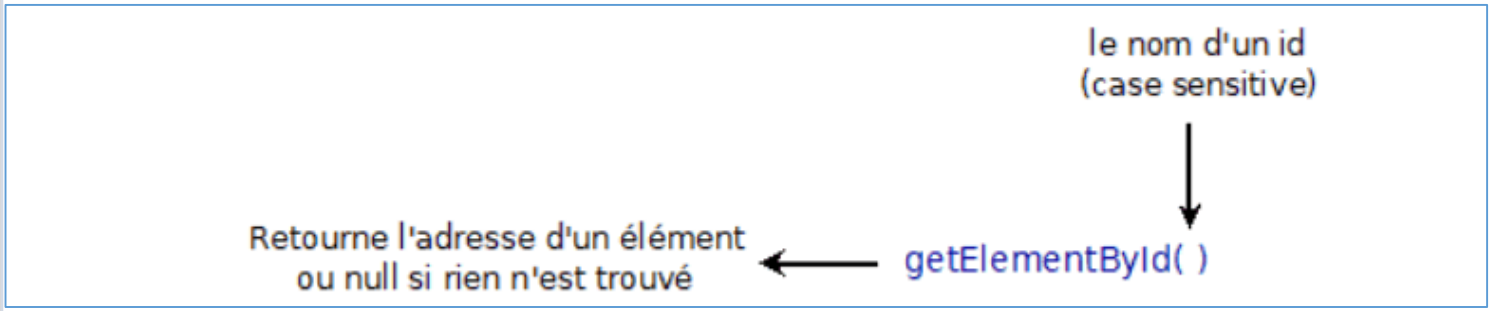
Exemple à tester

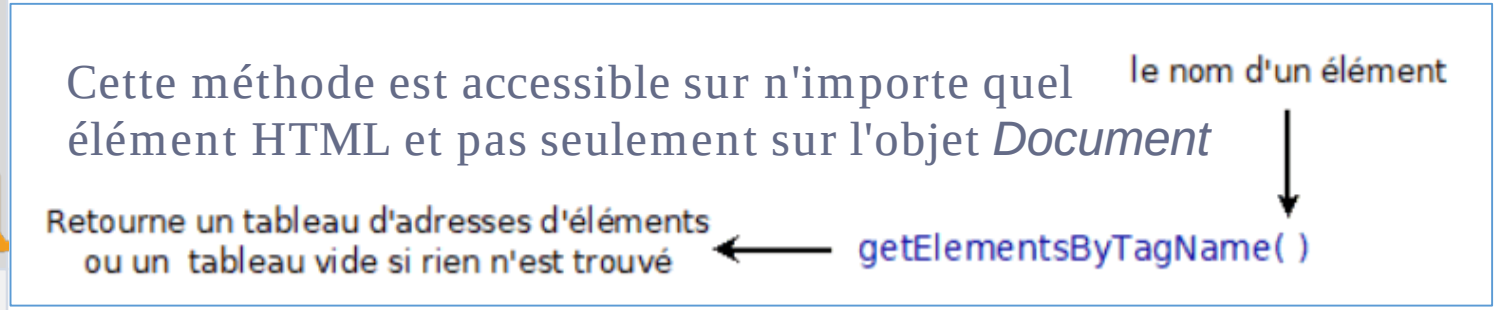
-
- Un peu de texte <a>et un lien
-
- <script>
- var a= document.getElementById('myDiv');
- alert(a.**innerHTML**);
- alert(a.**textContent**);
- </script>

Modifier le contenu d’un élément HTML

- La propriété innerHTML permet de modifier le contenu d’un élément HTML.
- il suffit d’utiliser innerHTML sur un élément et de lui affecter une nouvelle valeur.
- e.g.,
```
document.getElementById('myDiv').innerHTML =
'<p>Ceci est un autre paragraphe</p>';
```
- Pour ajouter du contenu, et ne pas modifier le contenu déjà en place, il suffit d’utiliser **+=** à la place de l'opérateur d'affectation

Différentes méthodes de l'objet *document*



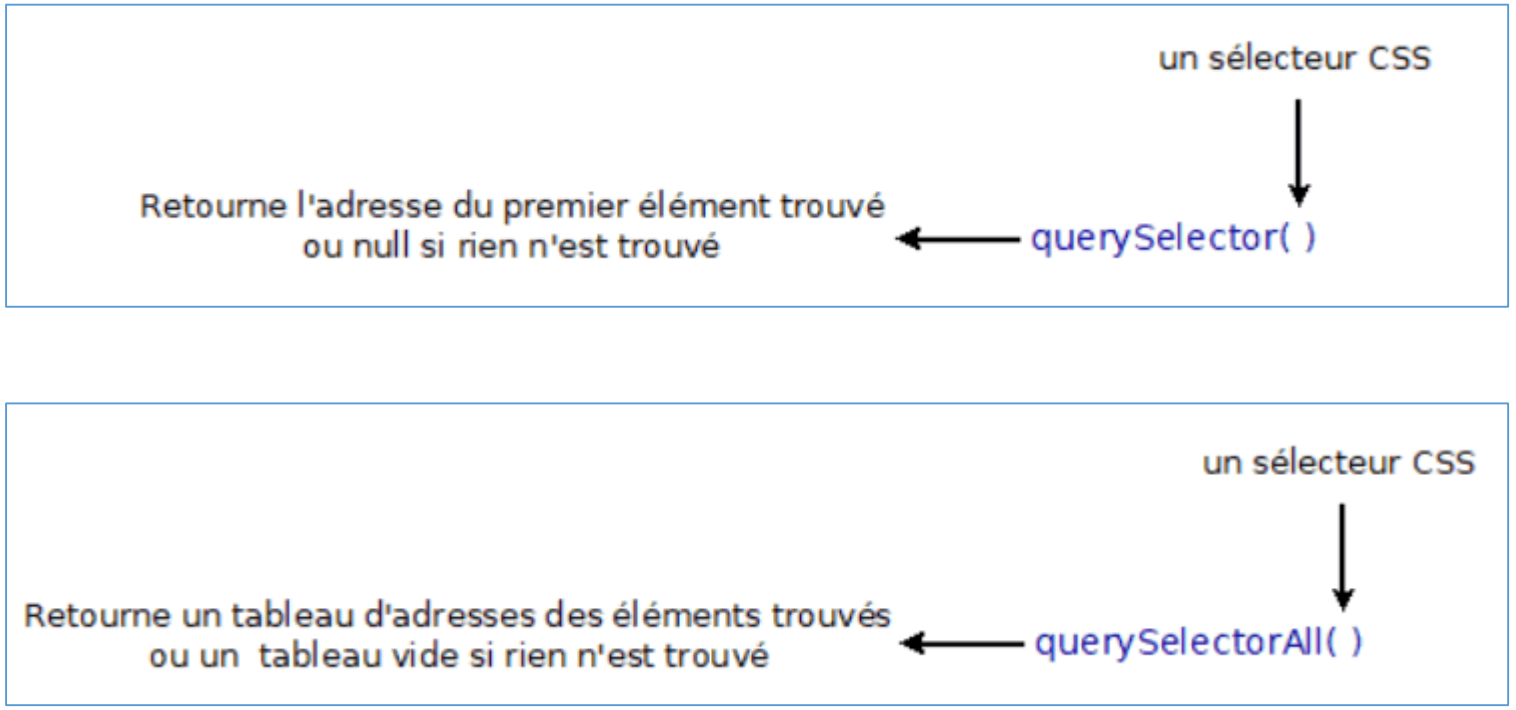



```
<div id="myDiv">
<p>Un peu de texte <a>et un
lien</a></p>
</div> <script>
var div =
document.getElementById('myDiv');
alert(div);
</script>
```

```
var divs =
document.getElementsByTagName('
div');
for (var i = 0, c = divs.length
; i < c ; i++)
{
alert('Element n° ' + (i + 1) +
' : ' + divs[i]);
}
```


Mohamed Karim Abdmouleh
2025

Différentes méthodes de l'objet *document*



Exemple à tester

```
<div id="menu">
        <div class="item">
                <span>Élément 1</span>
                <span>Élément 2</span>
        </div>
</div>
```

```
<script>
var query = document.querySelector('#menu .item span');
queryAll = document.querySelectorAll('#menu .item span');
alert(query.innerHTML);
alert(queryAll.length);
alert(queryAll[0].innerHTML + ' - ' + queryAll[1].innerHTML);
</script>
```


Mohamed Karim Abdmouleh
2025

Éditer les éléments HTML : les attributs

- Pour ajouter un attribut à un élément HTML, il existe deux manières
- **Avec la propriété attribute**

```
<img id="myImage" src="image1.gif" class="galerie">
<script>
document.getElementById("myImage").src = "image2.jpg";
</script>
```

- **Avec la méthode setAttribute()**

```
document.getElementById("myImage").setAttribute("src",
"image2.jpg");
```


Éditer les éléments HTML : les attributs

- **Exception avec l'attribut « class »**
- À la place de **class**, il faudra utiliser **className**

```
<img id="myImage" class="galerie">
<script>
document.getElementById("myImage").className = "images";
alert (document.getElementById("myImage").className)
</script>
```


Modifier le style CSS d'un élément HTML

- Il faut utiliser la propriété style de l'objet Element suivie de la propriété CSS à ajouter/modifier.

```
<p id="p1">Le style de ce paragraphe est modifié</p>
<script>
document.getElementById("p1").style.color = "red";
</script>
```


Ajouter et insérer des éléments HTML

Ajouter et insérer des éléments HTML avec DOM

1. Sélectionnez l'élément parent.
2. Créez un nouvel élément avec createElement().
3. Ajoutez du contenu à l'élément si nécessaire avec *textContent, innerHTML ou document.createTextNode*.
4. Ajoutez les attributs du nouvel élément avec setAttribute() si nécessaire.
5. Insérez l'élément dans le document avec appendChild() ou insertBefore().

appendChild va insérer un objet en tant que **dernier** enfant d'un autre objet.
La méthode **insertBefore(),** insère un objet juste avant un élément comme son nom l'indique.

Mohamed Karim Abdmouleh
2025

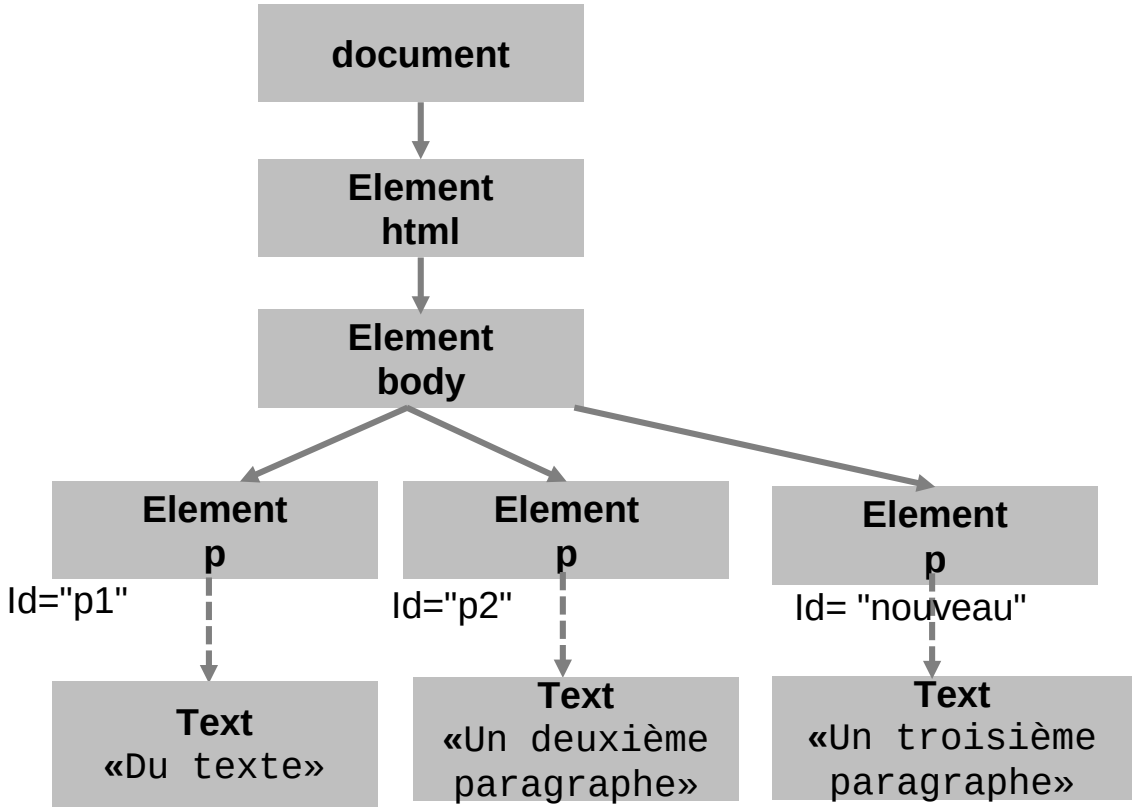
Exercice 1

- Ajouter, en utilisant le DOM, un troisième paragraphe au document suivant

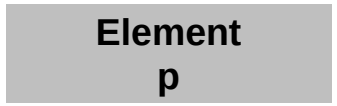
```
<body>
<p id="p1" class="para">Du texte </p>
<p id="p2" class="para">Un deuxième paragraphe</p>
</body>
```

Du texte

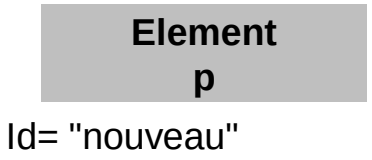
Un deuxième paragraphe



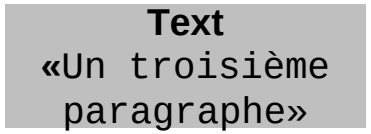
1 `var Pn = document.createElement('p');`



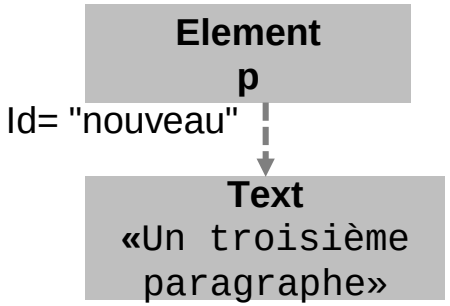
2 `Pn.id='nouveau';`



3 `var texte = document.createTextNode('Un troisième paragraphe');`



4 `Pn.appendChild(texte);`

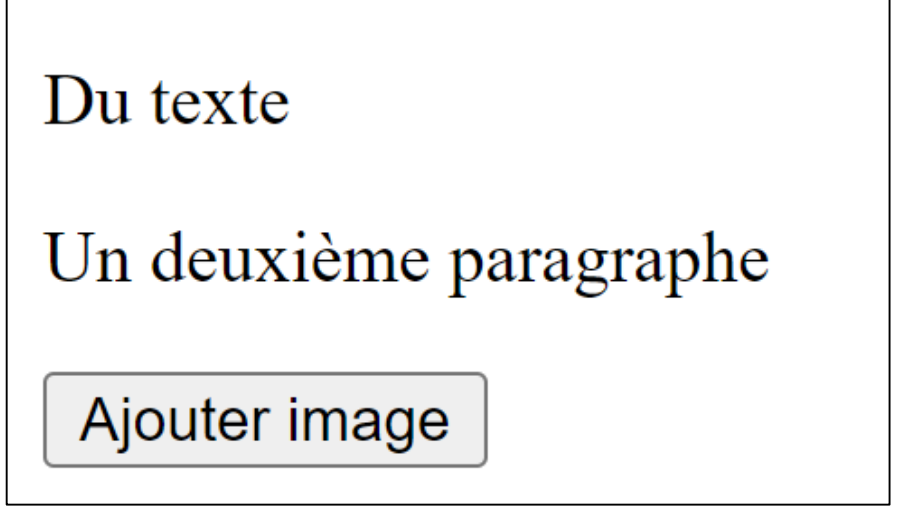


5 `document.body.appendChild(Pn);`

Correction

- <body>
- <h1>Le DOM</h1>
- Du texte
- Un deuxième paragraphe
- <script>
- //On crée un élément de type p
- **var Pn = document.createElement('p');**
- // On ajoute un attribut id à notre paragraphe
- **Pn.id='nouveau';**
- // On crée un noeud de type texte
- **var texte = document.createTextNode('Un troisième paragraphe');**
- // On insère le texte dans notre paragraphe
- **Pn.appendChild(texte);**
- //On insère notre élément en tant que dernier enfant de body
- **document.body.appendChild(Pn);**
- </script> </body>

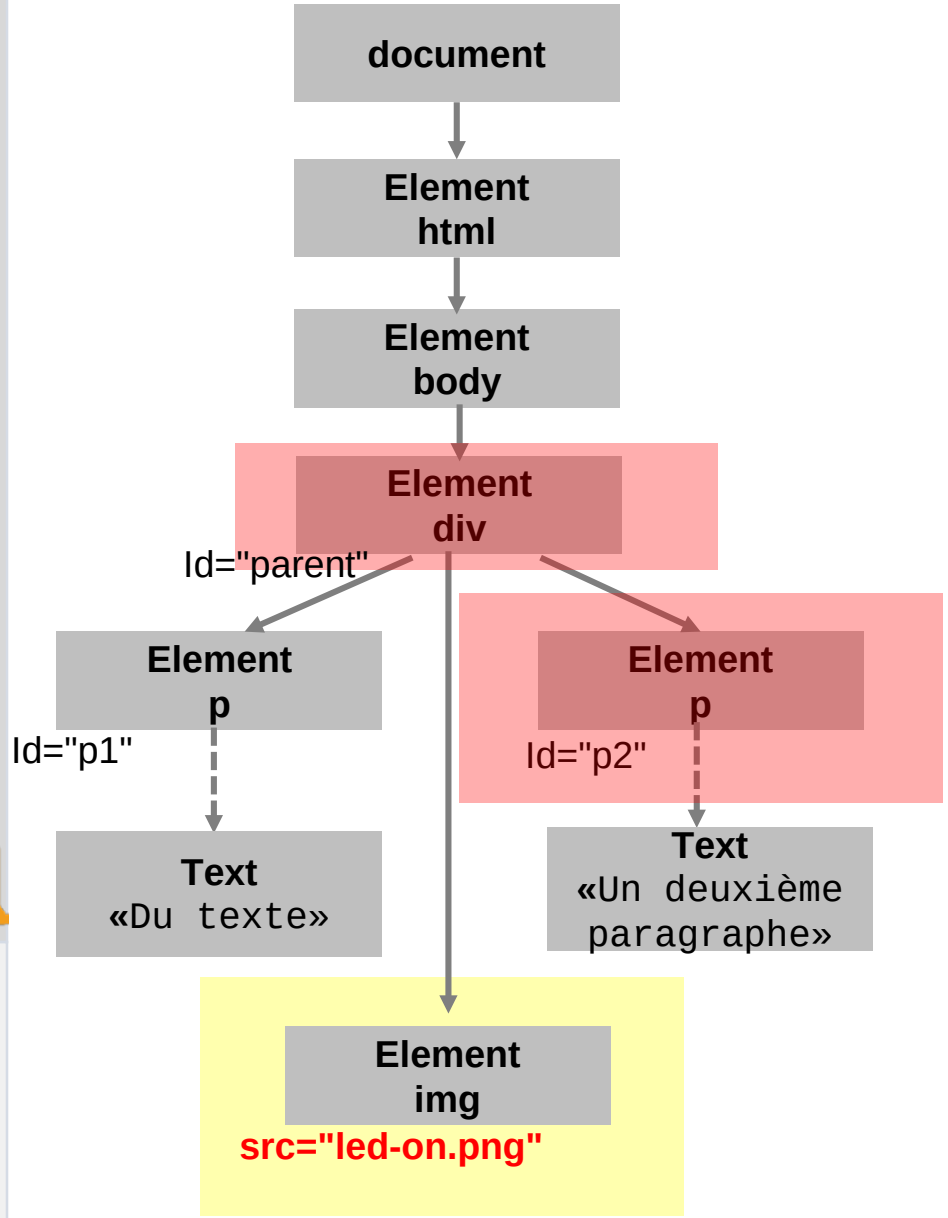
Exercice 2



- ajouter, en utilisant DOM, une image entre les deux paragraphes

```
<body>
<div id="parent">
    <p id="p1">Du texte </p>
    <p id="p2">Un deuxième paragraphe</p>
</div>
<button onclick="f()"> Ajouter image </button>
</body>
```


Mohamed Karim Abdmouleh
2025



1 `var Pn = document.createElement('img');`

Element
img

2 `Pn.src='led-on.png';`

Element
img
src="led-on.png"

3 `position= document.getElementById('p2');`

4 `parent= document.getElementById('parent');`

5 `parent.insertBefore(Pn,position);`

Correction

```
function f(){

    var Pn = document.createElement('img');
    position= document.getElementById('p2');
    parent= document.getElementById('parent');
    Pn.src='led-on.png';
    parent.insertBefore(Pn,position);

    }
```


Exercice 3

- ajouter, en utilisant le DOM, un nouveau paragraphe en première position

```
<body>
<h1>Le DOM</h1>
<p class="para">Du texte </p>
<p class="para">Un deuxième paragraphe</p>
</body>
```


Correction

- <script>
- //On crée un élément de type p
- **var Pn = document.createElement('p');**
- // On ajoute un attribut id à notre paragraphe
- **Pn.id='nouveau';**
- // On crée un noeud de type texte
- **var texte = document.createTextNode(' Un troisième paragraphe ');**
- // On insère le texte dans notre paragraphe
- **Pn.appendChild(texte);**
- // On accède à notre premier paragraphe
- **var P1=document.querySelector('.para'); // utiliser la console pour verifier le contenu de P1 (e.g., alert(P1))**
- //On insère le nouveau paragraphe juste avant
- **document.body.insertBefore(Pn,P1);**
- </script>

Remarque

- vous pouvez utiliser **innerHTML** pour ajouter du contenu HTML à un élément existant sans avoir à créer de nouveaux éléments avec **createElement().** Cependant, l'utilisation de innerHTML écrase le contenu existant de l'élément,

Modifier ou supprimer des éléments HTML

Modifier ou supprimer des éléments HTML

- **Supprimer un élément HTML**
- La méthode **removeChild**() : prend le nom de l'élément à retirer en argument.
- Cette méthode supprime un élément HTML enfant ciblé relativement à son parent. Il faut appliquer cette méthode à partir de l'élément parent.
- parent.removeChild(child);

```
<body> <p class="para" id="pp">Du texte </p>
… </body>
 document.body.removeChild(document.getElementById("pp"))
```


Modifier ou supprimer des éléments HTML

- **Remplacer des éléments HTML**
- la méthode replaceChild() : cette méthode prend deux arguments en entrée (la valeur de remplacement et le noeud qui doit être remplacé)
- **parent.replaceChild(nouvelElement, element);**
- Un deuxième paragraphe
- ….
- var Pn2 = document.createElement('p');
- Pn2.id='nouveau2';
- var texte2 = document.createTextNode(' Un 4eme paragraphe ');
- Pn2.appendChild(texte2);
- **document.body.replaceChild(Pn2, document.getElementById("pp2"));**

Modifier ou supprimer des éléments HTML

- La méthode **cloneNode()** sert à cloner un élément.
- Elle requiert un paramètre booléen (true ou false)
  - true: clonage avec les enfants et les attributs.
  - false: clonage sans enfants et sans attributs.

```
var paragraph1 = document.getElementById("pp");
var paragraph2 = paragraph1.cloneNode(true);
document.body.appendChild(paragraph2);
```


Mohamed Karim Abdmouleh
2025

Propriétés et méthodes des éléments

- **innerHTML :** non standard (mais implémenté partout), code HTML interne de l'élément
- **textContent :** contenu textuel de l'élément (tout le texte, en ignorant les balises). Pour IE: à partir v9
- **parentNode :** nœud parent
- **childNodes :** tableaux des nœuds fils. Chaque nœud fils est, soit un nœud élément (de nodeType 1), soit un nœud texte (nodeType 3)
- **firstChild :** équivalent de childNodes[0]
- **lastChild :** dernier fils
- **nodeName :** nom de la balise
- **nodeType :** La propriété nodeType fournit une autre méthode “à l’ancienne” pour obtenir le “type” d’un nœud DOM.

Il a une valeur numérique :

  - elem.nodeType == 1 pour les nœuds élément (balises),
  - elem.nodeType == 3 pour les nœuds texte,
  - elem.nodeType == 9 pour l’objet document,

- **id :** identifiant du nœud
- **className :** classe (au sens CSS) de l'élément
- **style :** accès aux propriétés CSS de l'élément
- **getAttribute(NOM) :** valeur d'un de ses attributs. Ne fonctionne pas pour l'attribut "class" sous IE.
- **setAttribute(NOM,VALEUR) :** pour un nœud/élément, permet de fixer la valeur d'un attribut. Ne fonctionne pas sous IE pour les styles css ni pour l'attribut "class"

Mohamed Karim Abdmouleh
2025

Récap

Node & HTMLElement

« Propriétés »

```
element.className
element.innerHTML
element.textContent
element.nodeName
element.nodeType
element.style
```

Arbre DOM

```
element.parentNode
element.children
element.childNodes
element.insertBefore()
element.removeChild()
```

```
element.addEventListener()
```

Chercher éléments

```
element.getElementsByClassName()
element.getElementsByTagName()
element.querySelector()
element.querySelectorAll()
```

Les évènements et le DOM

Les évènements et le DOM

- La méthode JavaScript **addEventListener**()
- Cette méthode appartient à l'objet Element.
- Deux arguments pour fonctionner : le nom de l'évènement déclencheur de l'action et le code relatif à l'action à effectuer.
- La méthode addEventListener() permet de:
    - lier plusieurs évènements différents à un même élément HTML
    - créer de multiples événements identiques pour un même élément

Différence entre *OnEvenement* et *addEventListener*

```
<body>
<input type="button" class="btn" value="Cliquer ici">
</body>
<script>
document.querySelector('.btn').onclick=function(){
console.log("Hello");};
document.querySelector('.btn').onclick=function(){
console.log("world");};
</script>
```

?

Différence entre *OnEvenement* et *addEventListener*

```
<body>
<input type="button" class="btn" value="Cliquer ici">
</body>
<script>
document.querySelector('.btn').addEventListener('click',function(){
console.log("Hello"); });
document.querySelector('.btn').addEventListener('click',function(){
console.log("world");});
</script>
```

?