IIT
INSTITUT
INTERNATIONAL
TECHNOLOGIE
Université Nord Américaine Privée

Programmation web II




Fatma Ellouze

fatma.ellouze@iit.ens.tn

1ère année Génie Informatique

(Version 0)

Retour sur le langage JavaScript

La Console de JS

- La console est **un outil indispensable lorsque l'on souhaite écrire quelques lignes de JavaScript**
- On peut exécuter du JavaScript dans la console web du navigateur
- La console du navigateur **fournira des informations sur les erreurs de page web**.
- Pour ouvrir la console dans la plupart des navigateurs, on peut faire :
  - un clic droit sur la souris. Cliquer sur **Inspecter** l'élément.
  - **Ou Ctrl + Shift + J / Ctrl + Shift + i**
- console.log est **une méthode qui permet d'afficher le contenu de variables à différents moments**. Il permet aussi d'informer publiquement les autres développeurs de ce que fait le code.

Où placer le javascript?

1. Directement dans le html (A la fin du body plutôt que dans le head pour accélérer l'exécution)

```
<script> codeJS</script>
```

2. Dans l'action d'un button: Event

```
<button  onclick="codeJS">  Ok  </button>
<p id="idTxt"> lorem….. </p>
<button
onclick="document.getElementById('idTxt').innerText='Bonjour'"> Cliquer
</button>
```

3. Dans un fichier.js à part

```
<script src="js/tp1.js"></script>
```

Changer le HTML en JS

- Lorem…
- 1) Récupération d’un élément à partir de son id getElementById()
- 2) Modification d’une propriété de l’élément innerText,.innerHtml, .src, .style…

- Exemple Changement propriété image
- <img id="idImg">
- …
- document.getElementById(’idImg').src=’iit.png’;

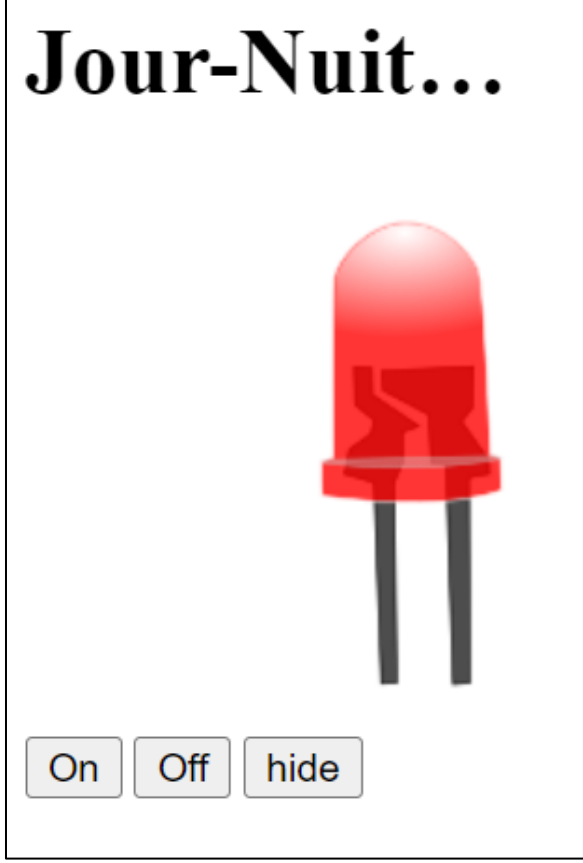
Changement style css

- **Changement de proprietes**
- document.getElementById(‘idTxt').style.fontSize=‘25px’;
- document.getElementById(‘idTxt').style.color=‘red’;
- document.getElementById(‘idTxt').style.backgroundColor=‘red’;

- **Changement de la propriete: display**
- // cacher
- document.getElementById(‘idTxt').style.display=‘none’;
- // afficher
- document.getElementById(‘idTxt').style.display=‘block’;

Exemple : LED

- Mettre un bouton [On/off] qui allume/éteint la Led
- Ajoutez un bouton [hide] qui, lorsqu'il est enfoncé, rend la LED invisible. Ensuite, lorsqu'il est à nouveau enfoncé, il la fait réapparaître.



Notions Fondamentales en JS

- Nommage: maVariable, maFonction, MaClasse
- Déclaration non typé
- var x = true; // boolean
- var entier= [11, 45, 17]; // Tableau
- var personne= {prenom: "Fatma", nom: "Ellouze"}; // Objet
- var prenom= "Fatma"; ou var prenom= ‘Fatma’; // Chaine de caractères
- Variable globale On ne met pas le mot **var** Exemple x=12;
- if(a===b) // vérifie également que les types sont identique : A UTILISER

Affichage en js

1. Dans la console (console.log(prenom);)

2. Dans une popup window (window.alert(prenom) ;)

3. Dans le html

```
document.write("Hello World");
```

Ou

```
document.getElementById("txtHello").innerText= "Hello World";
```


Utiliser Bibliothèque extérieure : exemple de l’objet Date

- var d = newDate(2012,2,29);
- var y= d.getFullYear() ; 2000
- var m= d.getMonth() ; 2

La gestion des erreurs

- try{
- ….
- window.alert("Hello");
- Sdfgdfghjkl; // c’est l’erreur

- }catch(erreur){
- // je capture l’erreur et la gère
- window.alert(erreur.message);
- throw("Stop !!!");
- }
- window.alert("Et ça continue encore et …");

Fonction en JS : 2 déclarations possibles

- function nomFonction(){
- … // le code de la fonction
- }
- Ou fonction **anonyme** Cette syntaxe crée une fonction sans nom et l'assigne à une variable nomFonction.
- var nomFonction = function() {
- … // le code de la fonction
- }
- // l'appel…
- nomFonction();

Les écouteurs d'Event

Exemples

- **mouse(souris):**
  - click: au clic sur un élément
  - mouseenter: la souris passe par dessus la zone qu'occupe un élément
  - mouseleave: la souris sort de cette zone
- **keyboard(clavier)**
  - keydown: une touche du clavier est enfoncée
  - keyup: une touche a été relachée
- **window(fenêtre)**
  - scroll: défilement de la fenêtre
  - resize: redimensionnement de la fenêtre
- **form(formulaires)**
  - change: pour les éléments <input>, <select> et <textarea>, quand l'utilisateur change une de leurs valeurs
  - submit: à l'envoi d'un formulaire
- **document**
  - DOMContentLoaded: lancé quand le document HTML est complètement chargé et analysé sans attendre que les images et les CSS soient chargés
  - Load The browser has finished loading the page

Gestion des event

- ***AVANT*** on mettait directement l'evt et l'appel de la fct dans le .html
- <button **onclick**="displayDate()">The time is?</button>
- ***MAINTENANT*** on utilise les **EventListener**
- <!------index.html---------->
- <button id="butId">Resize</button>
- ----------main.js-------------------
- // recup des id
- var but= document.getElementById("butId");
- // Association des événements avec addEventListener
- but.addEventListener("click", resize);
- // def des fct
- function resize(){
- … }

Si on veut récupérer la source d’un évènement

- form.addEventListener("click", function(**e**) {
- // Récupérer la source de l'événement
- var source = **e.target**;

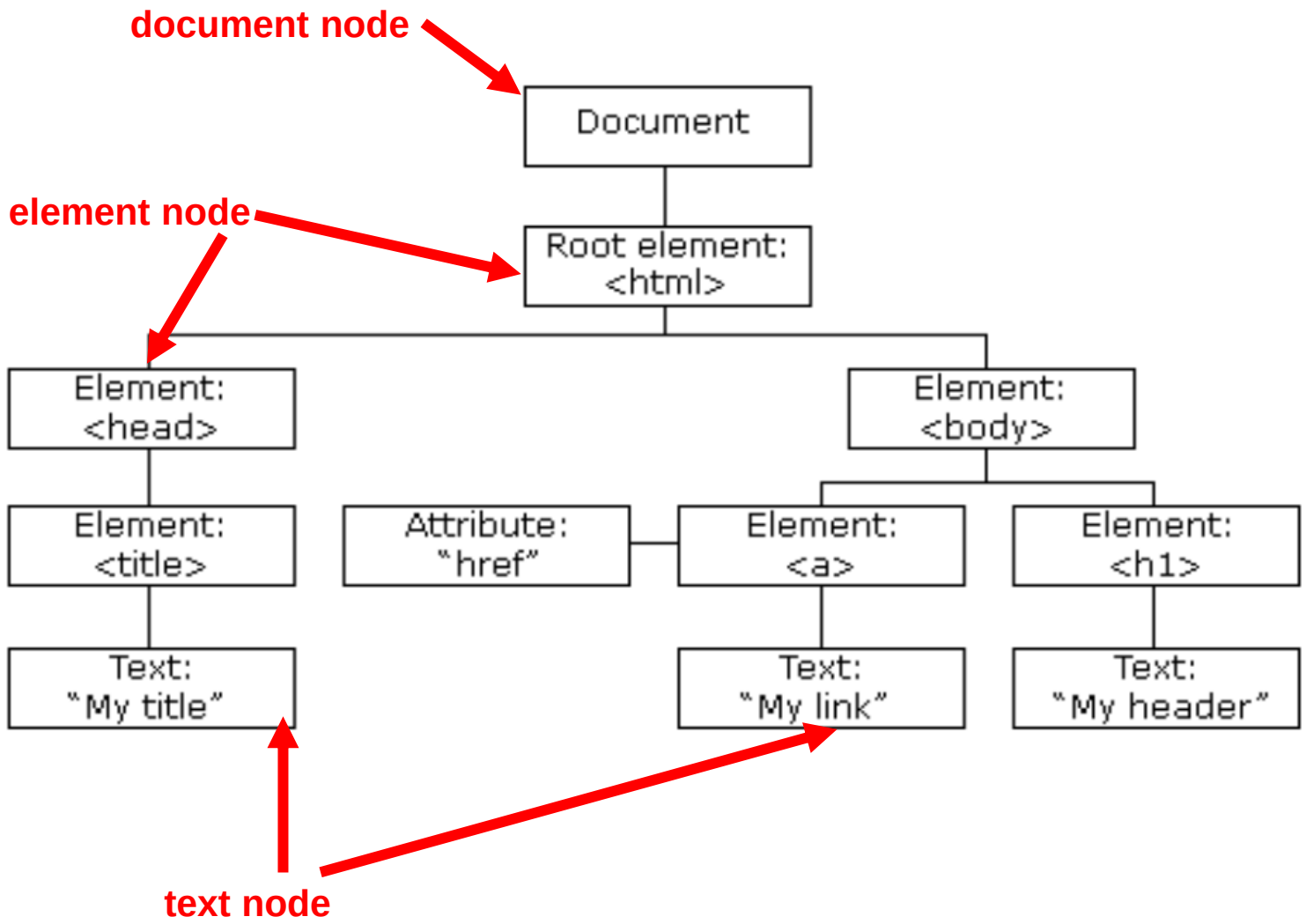
- console.log("La source de l'événement est :", source);
- });

DOM = Document Object Model

- Le DOM est une interface de programmation (API) qui représente la structure d'un document HTML ou XML sous forme d'un arbre.
- Il fournit une manière de **manipuler la structure**, le **style** et le **contenu** d'une page web en utilisant **JavaScript**.
- Le DOM est une **spécification indépendante du langage** et peut être **manipulé avec différents langages** de programmation,
- 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>
```



Remarque chaque nœud DOM se transforme en un objet JS

Les **propriétés** des nœuds **Element**

- Tout **élément** HTML DOM possède des propriétés auquel on peut accéder et/ou modifier avec **JavaScript**, exemples :
- x.innerHTML – la valeur du contenu des éléments HTML dans le noeud x
- x.textContent – 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

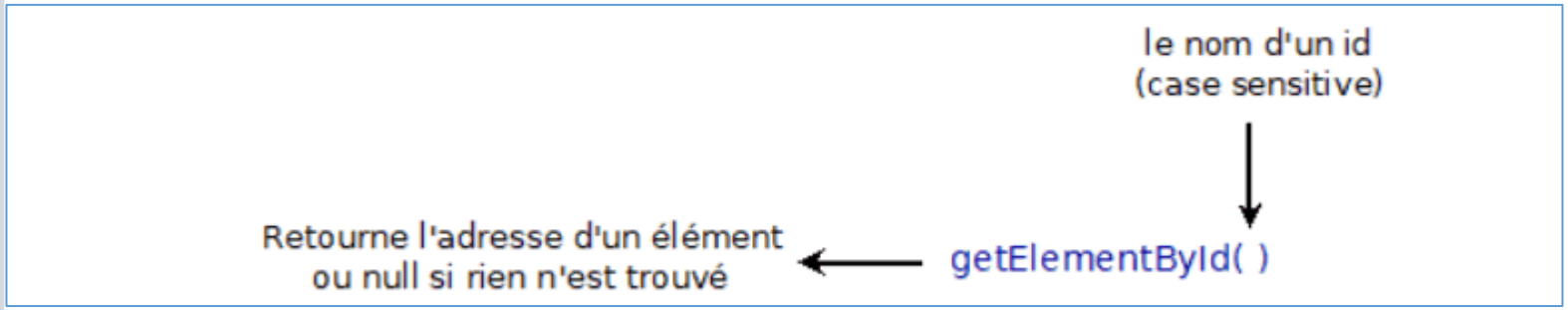
Exemple à tester

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

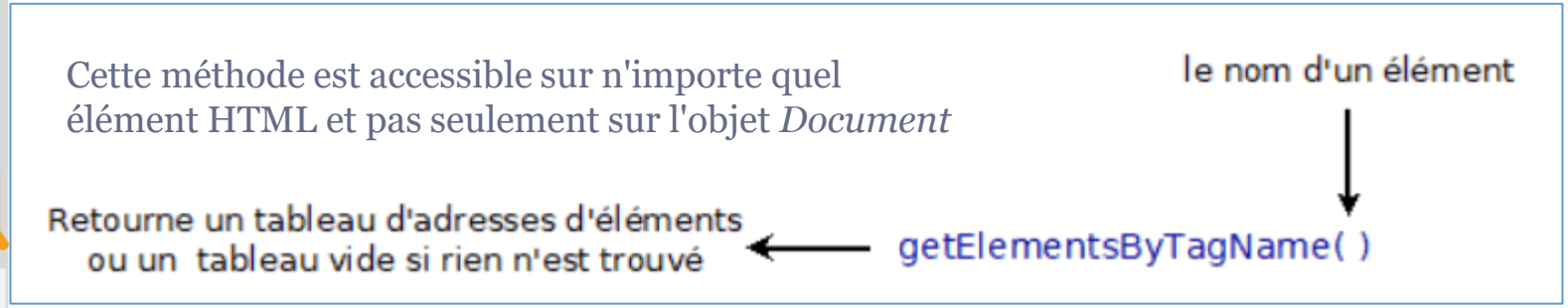
Modifier le contenu d’un élément HTML

- 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*

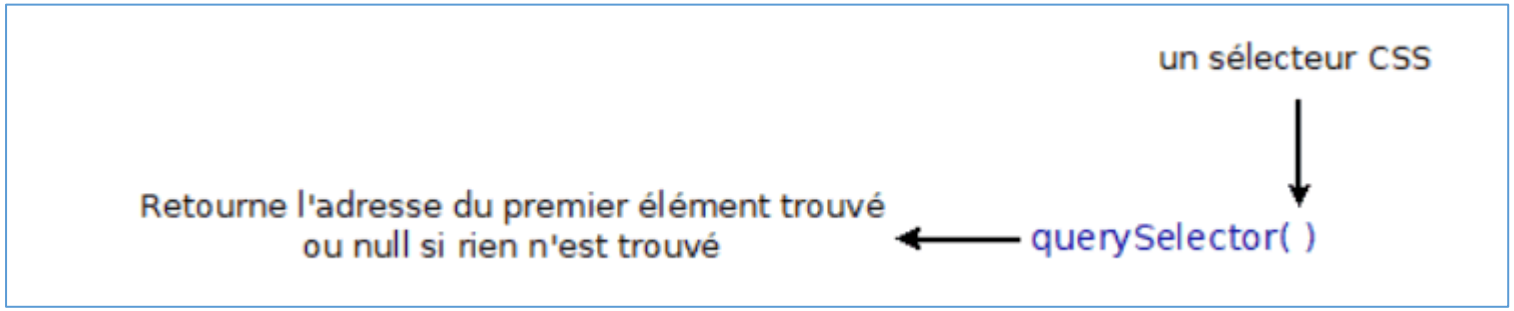


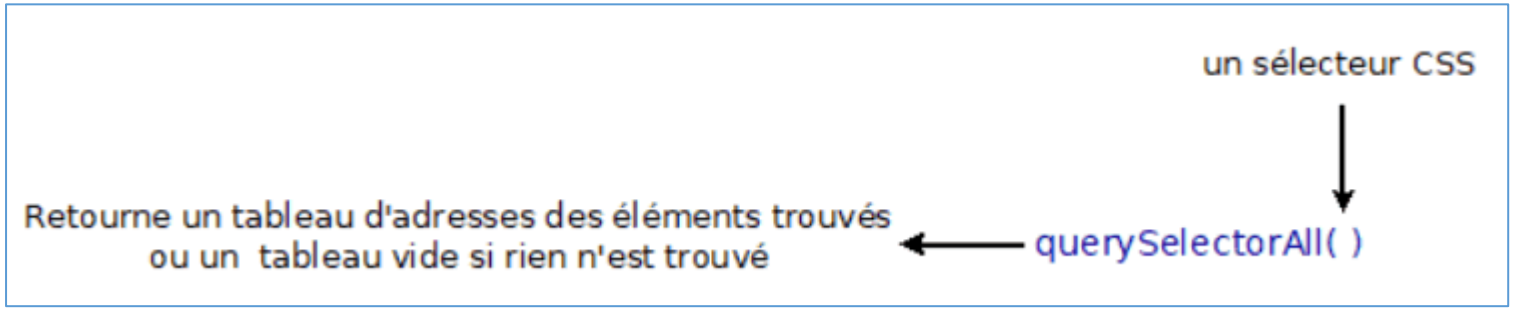

```
<div id="myDiv">
<p>Un peu de texte <a>et un lien</a></
</div> <script>
var div =
document.getElementById('myDiv');
alert(div);
</script>
```




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

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>
```


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>
```


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.

insérer des éléments HTML avec appendChild

```
// Étape 1 : Sélectionner l'élément parent
var parentElement = document.getElementById("parent");

// Étape 2 : Créer un nouvel élément
var newElement = document.createElement("div");

// Étape 3 : Ajouter du contenu à l'élément
newElement.textContent = "Nouvel élément ajouté";

// Étape 4 : Configurer les attributs de l'élément
newElement.setAttribute("id", "new");

// Étape 5 : Insérer l'élément dans le document
parentElement.appendChild(newElement);

// Étape 6 : Actualiser le document (pas nécessaire car JavaScript s'exécute en temps réel)
```


insérer des éléments HTML avec insertBefore

```
<body>
<div id="container">
  <p id="existingElement">Je suis un élément existant.</p>
</div>
<script>
// Création d'un nouvel élément
var newElement = document.createElement("p");
var newText = document.createTextNode("Je suis un nouvel élément.");
// Ajout du texte à l'élément
newElement.appendChild(newText);
// Récupération de l'élément existant
var existingElement = document.getElementById("existingElement");
// Récupération du parent de l'élément existant
var parentElement = existingElement.parentNode;
// Insertion du nouvel élément avant l'élément existant
parentElement.insertBefore(newElement, existingElement);
</script>   </body>
```


Ajouter et insérer des éléments HTML avec DOM

- **Créer un nouvel élément HTML**
- Utiliser la méthode **createElement**() de l'objet Document.
- Cette méthode prend en argument le nom de l'élément HTML à créer.

```
<body>
<h1>Le DOM</h1>
<p class="para">Du texte </p>
<p class="para">Un deuxième paragraphe</p>
<script>
//On crée un élément de type p
var x = document.createElement('p');
</script>
</body>
```


Ajouter et insérer des éléments HTML

- Pour ajouter du texte : la méthode **createTextNode**() qui, comme son nom l'indique, va créer un nouveau noeud de **type texte**.
- `var node = document.createTextNode("ceci est le contenu du nouveau paragraphe.");`

Ajouter et insérer des éléments HTML

- **<u>Insérer du texte et un élément dans une page HTML</u>**
- La méthode **appendChild()** va insérer un objet en tant que **dernier** enfant d'un autre objet.
- Cette méthode prend le nom de l'objet à insérer en argument.
- exemples : x.appendChild(node);
- document.body.appendChild(x);…
- **<u>Insérer un élément HTML à un endroit précis</u>**
- La méthode **insertBefore(),** insère un objet juste avant un élément comme son nom l'indique.
- Deux arguments pour cette méthode : **l'élément à insérer** et **l'élément avant lequel il sera inséré**.

Exercice

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

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


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

- 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);
```