Ajouter, modifier ou supprimer des éléments du DOM avec JavaScript

Nous allons pouvoir utiliser ces connaissances pour ajouter, modifier, remplacer ou supprimer des nœuds dans le DOM.

Créer de nouveaux nœuds et les ajouter dans l'arborescence du DOM

On va pouvoir, en JavaScript, ajouter des éléments dans notre document. Pour cela, il va falloir procéder en deux étapes : on va déjà créer un nouveau nœud puis on va ensuite l'insérer à une certaine place dans le DOM.

Créer un nœud Element ou un nœud Text

Pour créer un nouvel élément HTML en JavaScript, nous pouvons utiliser la méthode `createElement()` de l'interface `Document`.

Cette méthode va prendre en argument le nom de l'élément HTML que l'on souhaite créer.

```
let newP = document.createElement('p');
```

Nous avons ici créé un nouvel élément `p`. Celui-ci ne contient pour le moment ni attribut ni contenu textuel, et n'a pas encore été inséré à l'intérieur de notre page à un endroit précis.

Pour insérer du texte dans notre nœud élément, on va pouvoir par exemple utiliser la propriété `textContent`.

```
let newP = document.createElement('p');

newP.textContent = 'Paragraphe créé et inséré grâce au JavaScript';
```

On peut également créer directement un nœud de type texte en utilisant la méthode `createTextNode()` de `Document` et ensuite insérer ce nœud dans un nœud élément avec l'une des méthodes que nous allons voir immédiatement.

```
let newP = document.createElement('p');
let newTexte = document.createTextNode('Texte écrit en JavaScript');

newP.textContent = 'Paragraphe créé et inséré grâce au JavaScript';
```

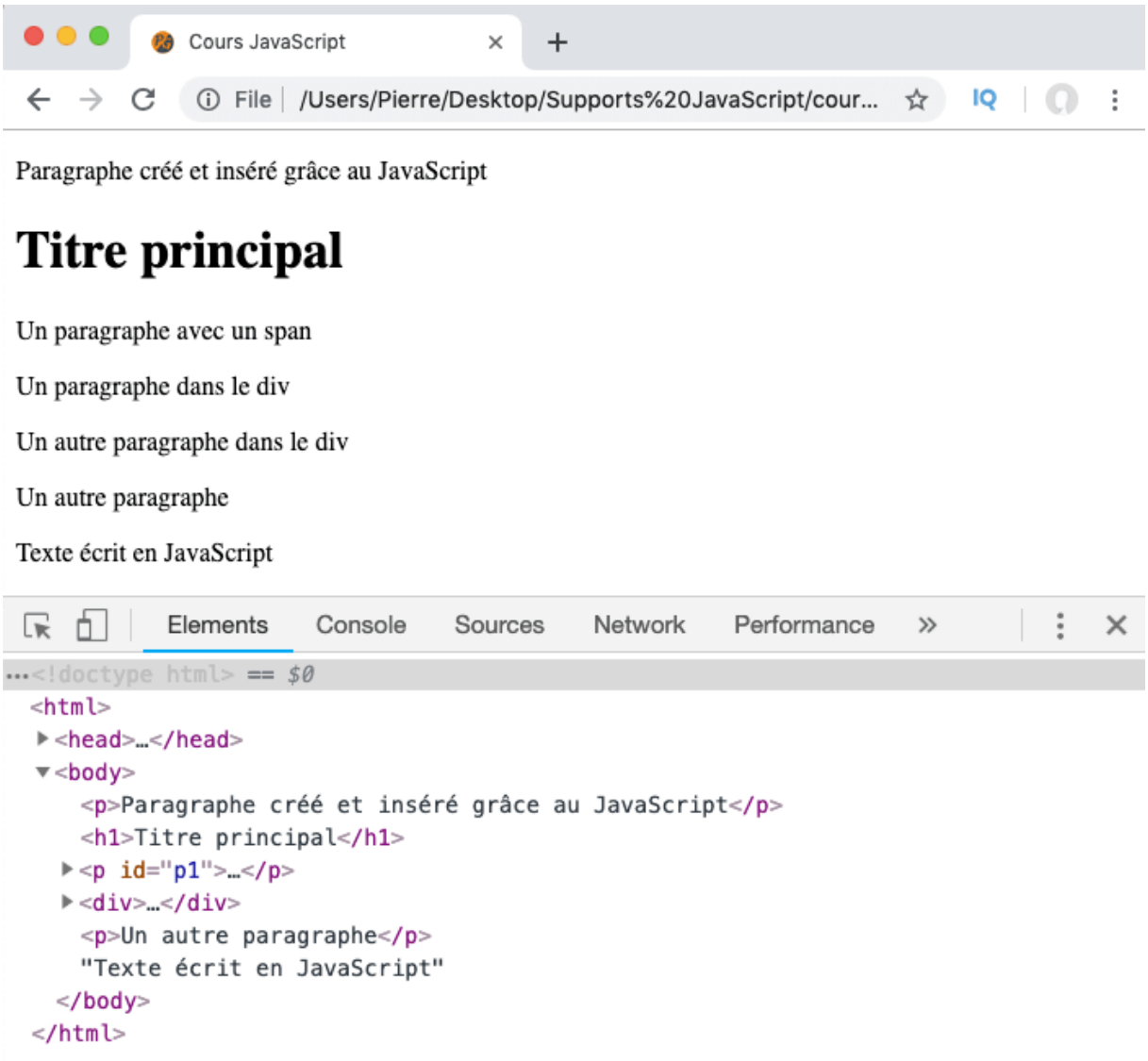
Insérer un nœud dans le DOM

Il existe différentes méthodes qui nous permettent d'insérer des nœuds dans d'autres nœuds. La différence entre ces méthodes va souvent consister dans la position où le nœud va être inséré.

Nous pouvons déjà utiliser les méthodes `prepend()` et `append()` du mixin `ParentNode`. Ces deux méthodes vont respectivement nous permettre d'insérer un nœud ou du texte avant le premier enfant d'un certain nœud ou après le dernier enfant de ce nœud.

```
<!DOCTYPE html>
<html>
    <head>
        <title>Cours JavaScript</title>
        <meta charset="utf-8">
        <link rel="stylesheet" href="cours.css">
        <script src='cours.js' async></script>
    </head>

    <body>
        <h1>Titre principal</h1>
        <p id='p1'>Un paragraphe <span>avec un span</span></p>
        <div>
            <p id='p2'>Un paragraphe dans le div</p>
            <p>Un autre paragraphe dans le div</p>
        </div>
        <p>Un autre paragraphe</p>
    </body>
</html>
```

```
let b = document.body;
let newP = document.createElement('p');
let newTexte = document.createTextNode('Texte écrit en JavaScript');

newP.textContent = 'Paragraphe créé et inséré grâce au JavaScript';

//Ajoute le paragraphe créé comme premier enfant de l'élément body
b.prepend(newP);

//Ajoute le texte créé comme dernier enfant de l'élément body
b.append(newTexte);
```

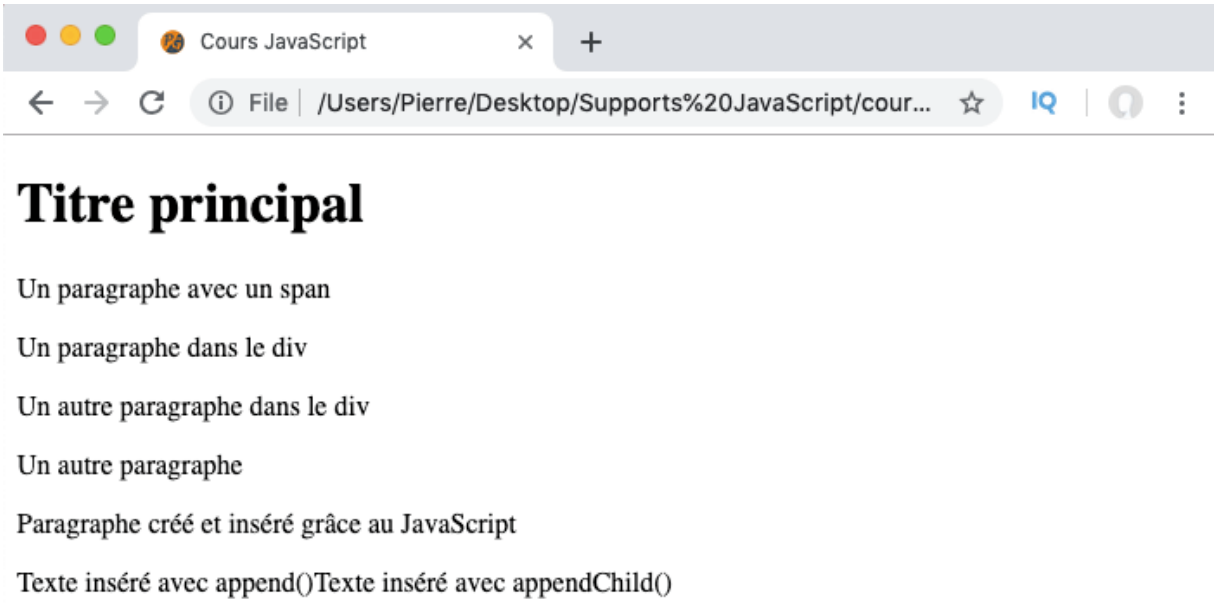


On peut également utiliser la méthode `appendChild()` de l'interface `Node` qui permet d'ajouter un nœud en tant que dernier enfant d'un nœud parent.

Les différences entre `append()` de `ParentNode` et `appendChild()` de `Node` sont les suivantes :

- La méthode `append()` permet également d'ajouter directement une chaine de caractères tandis que `appendChild()` n'accepte que des objets de type `Node` ;
- La méthode `append()` peut ajouter plusieurs nœuds et chaines de caractères au contraire de `appendChild()` qui ne peut ajouter qu'un nœud à la fois ;
- La méthode `append()` n'a pas de valeur de retour, tandis que `appendChild()` retourne l'objet ajouté.

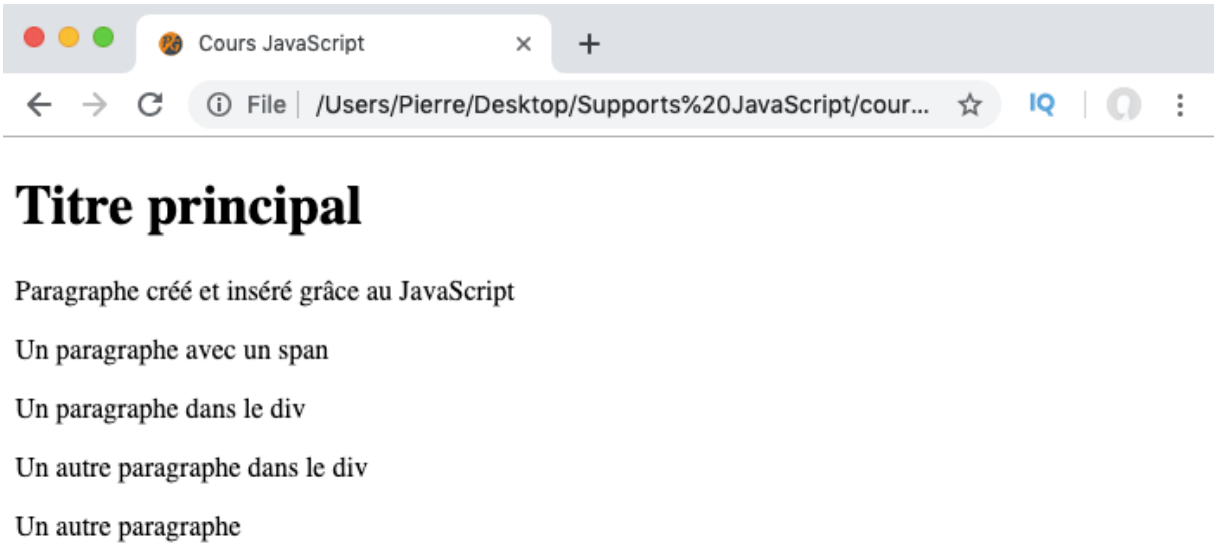
```
let b = document.body;
let newP = document.createElement('p');
let newTexte = document.createTextNode('Texte inséré avec appendChild()');

newP.textContent = 'Paragraphe créé et inséré grâce au JavaScript';

//Ceci fonctionne
b.append(newP, 'Texte inséré avec append()');

//Ceci fonctionne
b.appendChild(newTexte);

//Ceci ne fonctionne pas : b.appendChild('Texte écrit en JavaScript');
//Ceci ne fonctionne pas non plus : b.appendChild(newP, newTexte);
```



On peut encore utiliser la méthode `insertBefore()` de l'interface `Node` qui permet pour sa part d'insérer un nœud en tant qu'enfant d'un autre nœud juste avant un certain nœud enfant donné de ce parent.

Cette méthode va prendre en arguments le nœud à insérer et le nœud de référence c'est-à-dire le nœud juste avant lequel le nœud passé en premier argument doit être inséré.

```
let b = document.body;
let p1 = document.getElementById('p1');
let newP = document.createElement('p');

newP.textContent = 'Paragraphe créé et inséré grâce au JavaScript';

b.insertBefore(newP,p1);
```



Finalement, nous pouvons aussi utiliser les méthodes `insertAdjacentElement()`, `insertAdjacentText()` et `insertAdjacentHTML()` de l'interface `Element` pour insérer nos nœuds dans le DOM.

La méthode `insertAdjacentElement()` insère un nœud élément à une position donnée par rapport à l'élément sur lequel il est appelé.

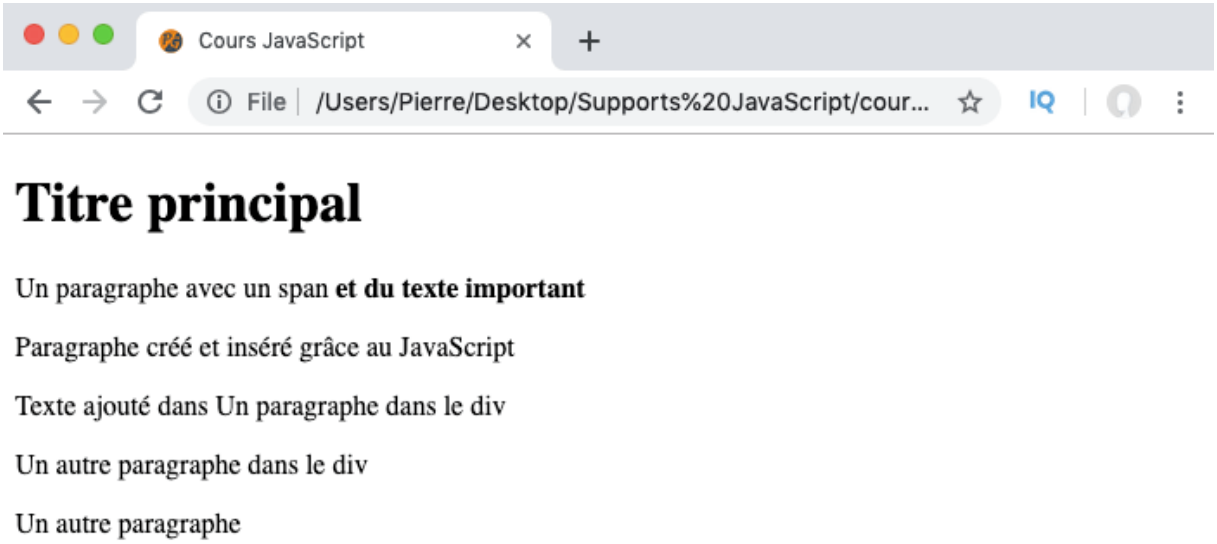
La méthode `insertAdjacentText()` insère un nœud texte à une position donnée par rapport à l'élément sur lequel il est appelé.

La méthode `insertAdjacentHTML()` analyse une chaine de caractères en tant que HTML et insère les nœuds créés avec le balisage donné dans le DOM à une certaine position spécifiée.

Pour chacune de ces trois méthodes, nous allons devoir spécifier la position où on souhaite insérer nos nœuds ainsi que le nœud à insérer en arguments. Pour la position, il faudra fournir l'un des mots clefs suivants :

- beforebegin : Insère le ou les nœuds avant l'élément ;
- afterbegin : Insère le ou les nœuds avant le premier enfant de l'élément ;
- beforeend : Insère le ou les nœuds après le dernier enfant de l'élément ;
- afterend : Insère le ou les nœuds après l'élément.

```
let b = document.body;
let p1 = document.getElementById('p1');
let p2 = document.getElementById('p2');
let newP = document.createElement('p');
let htmlContent = '<strong> et du texte important</strong>';

newP.textContent = 'Paragraphe créé et inséré grâce au JavaScript';

//Ajoute un paragraphe après p1
p1.insertAdjacentElement('afterend', newP);

//Ajoute le contenu de htmlContent avant la balise fermante de p1
p1.insertAdjacentHTML('beforeend', htmlContent);

//Ajoute du texte après la balise ouvrante de p2
p2.insertAdjacentText('afterbegin', 'Texte ajouté dans ');
```



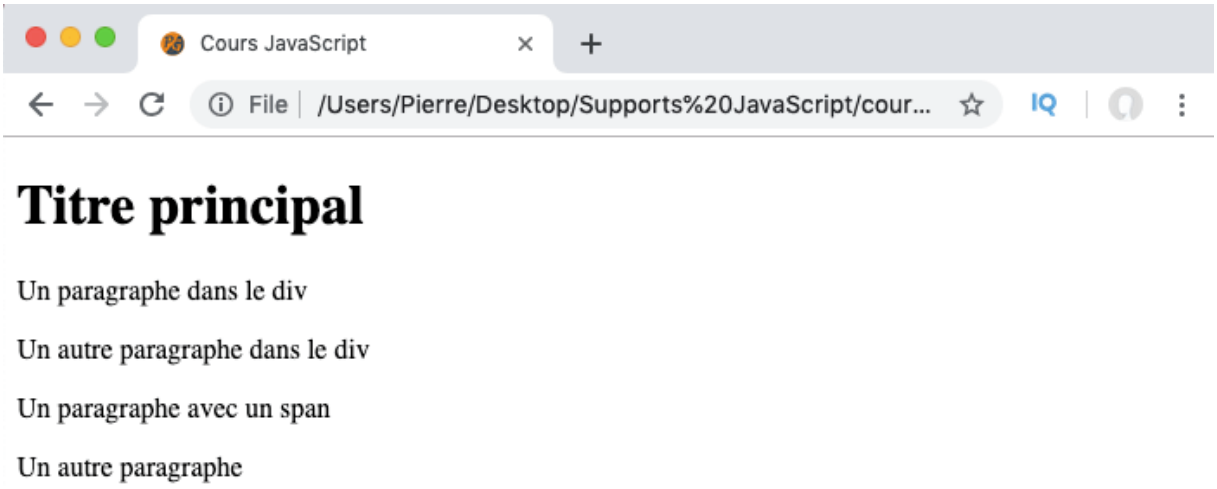
Note : le mixin `ChildNode` nous fournit également deux méthodes `before()` et `after()` qui permettent d'insérer un nœud avant ou après un certain enfant d'un certain nœud parent. Toutefois, ces deux méthodes sont récentes et ne sont donc pas encore supportées par tous les navigateurs.

Déplacer un nœud dans le DOM

Pour déplacer un nœud dans le DOM, on peut utiliser l'une des méthodes `appendChild()` ou `insertBefore()` de `Node` en leur passant en argument un nœud qui existe déjà et qui est déjà placé dans le DOM.

Dans ce cas-là, les méthodes vont déplacer le nœud dans le DOM vers la nouvelle position indiquée.

```
let b = document.body;
let p1 = document.getElementById('p1');
let p4 = b.lastElementChild; //On accède au dernier paragraphe

//On déplace p1 juste avant p4 dans le DOM
b.insertBefore(p1, p4);
```



Cloner ou remplacer un nœud dans le DOM

Pour cloner un nœud, on peut utiliser la méthode `cloneNode()` de `Node` qui renvoie une copie du nœud sur lequel elle a été appelée.

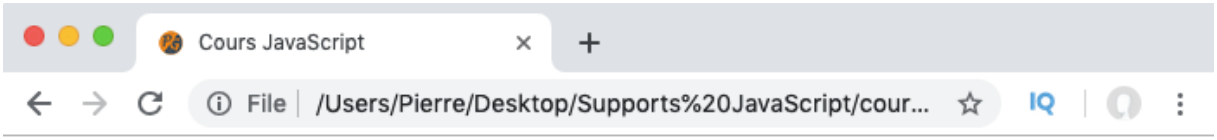
Cette méthode prend un booléen en argument. Si la valeur passée est `true`, les enfants du nœud seront également clonés. Si on lui passe `false`, en revanche, seul le nœud spécifié sera cloné.

Par défaut, cette méthode copie également le contenu du nœud cloné.

Pour remplacer un nœud, on utilisera plutôt la méthode `replaceChild()` de cette même interface qui va remplacer un certain nœud par un autre.

Cette méthode va prendre en arguments le nœud de remplacement et le nœud qui doit être remplacé. Notez que si le nœud de remplacement existé déjà dans le DOM, il sera d'abord retiré de son emplacement d'origine.

```
let b = document.body;
let p1 = document.getElementById('p1');
let p2 = document.getElementById('p2');
let p4 = b.lastElementChild; //On accède au dernier paragraphe
let newP = document.createElement('p'); //On crée un nouveau noeud
newP.textContent = 'Paragraphe créé et inséré grâce au JavaScript';


//On clone p1 et on insère le clone après p2
let cloneP1 = p1.cloneNode(true);
p2.insertAdjacentElement('afterend', cloneP1);

//On remplace p4 par newP
b.replaceChild(newP, p4);
```



Titre principal

Un paragraphe avec un span

Un paragraphe dans le div

Un paragraphe avec un span

Un autre paragraphe dans le div

Paragraphe créé et inséré grâce au JavaScript

Supprimer un nœud du DOM

Pour supprimer totalement un nœud du DOM, on peut déjà utiliser la méthode `removeChild()` de `Node` qui va supprimer un nœud enfant passé en argument d'un certain nœud parent de l'arborescence du DOM et retourner le nœud retiré.

On peut également utiliser la méthode `remove()` du mixin `ChildNode()` qui permet tout simplement de retirer un nœud de l'arborescence et qui dispose aujourd'hui d'un bon support par les navigateurs (cette façon de faire est plus récente que la précédente).

```
let b = document.body;
let p1 = document.getElementById('p1');
let p2 = document.getElementById('p2');

//Supprime p1 du DOM et renvoie le noeud supprimé
let eltDel = b.removeChild(p1);

//Supprime p2 du DOM
p2.remove()

alert('Noeud supprimé du DOM : ' + eltDel + '\nContenu : ' + eltDel.textContent);
```