Manipuler la page web avec le DOM

Objectif : Comprendre ce qu’est le DOM et comment sélectionner des éléments

Touchet Laurent / Afec 2025

Sommaire

C’est quoi le DOM ?

Le **DOM** (Document Object Model) est une représentation en mémoire de la page web.

Quand le navigateur charge une page HTML, il construit une sorte d’arbre avec tous les éléments de la page : balises, attributs, texte, etc.

Grâce à JavaScript, on peut **lire**, **modifier** ou **supprimer** des éléments de cette structure : c’est comme parler directement au navigateur.

Sélectionner des éléments du DOM

Sélectionner un seul élément du DOM :

```
document.getElementById("monId");
document.querySelector(".maClasse"); // ou body > .maClasse
```

Méthode	Description
getElementById("monId")	Sélectionne **un seul élément** qui a l’attribut id="monId"
querySelector("div.maClasse")	Sélectionne **le premier élément** qui correspond au **sélecteur CSS** donné

Sélectionner des éléments du DOM

Exemple de sélection d'un seul élément du DOM :

```
<script>
 let paragraphe = document.getElementById("message");
 paragraphe.innerText = "Bonjour modifié !";

 // Autre méthode :
 let memeParagraphe = document.querySelector("#message");
</script>
```

- **querySelector est plus souple :** il permet de cibler par classe, balise, combinaison CSS…

Sélectionner des éléments du DOM

Sélectionner plusieurs éléments du DOM :

```
document.getElementsByClassName("maClasse");
document.getElementsByTagName("li");
document.querySelectorAll("ul > li")
```

Méthode	Description
getElementsByClassName("maClasse")	Retourne une **HTMLCollection** (ressemble à un tableau)
getElementsByTagName("li")	Pareil mais par nom de balise
querySelectorAll("ul > li")	Retourne une **NodeList** avec tous les éléments correspondants

Modifier une propriété d'un objet JS

```
<ul>
 <li class="item">Pomme</li>
 <li class="item">Cerise</li>
</ul>
<script>
 let items = document.querySelectorAll(".item");
 items.forEach((li, index) => {
  li.innerText += " 🍒";
  li.style.color = "purple";
 });
</script>
```

Modifier un ou plusieurs éléments

Modifier le texte d’un élément

```
let element = document.querySelector(‘.maClasse’);
element.innerText = "Nouveau texte";
element.innerHTML = "<strong>Texte en gras</strong>";
```

Méthode	Description
element.innerText	Modifie le **texte affiché** à l’écran
element.textContent	Pareil, mais plus brut (affiche tout, même ce qui est caché)
element.innerHTML	Permet d’ajouter du **HTML dans l’élément** (Attention à la sécurité)

Modifier un ou plusieurs éléments

Modifier le style

```
let element = document.querySelector(‘.maClasse’);
element.innerText = "Nouveau texte";
element.innerHTML = "<strong>Texte en gras</strong>";
```

```
element.style.color = "red";
element.style.display = "flex";
element.style.backgroundColor = “#333333”
```

Astuce : le style est en camelCase (backgroundColor, pas background-color).

Modifier un ou plusieurs éléments

Modifier plusieurs éléments à la fois

```
<p class="info">Texte 1</p>
<p class="info">Texte 2</p>
<p class="info">Texte 3</p>
```

```
let infos = document.querySelectorAll(".info");
infos.forEach((p, index) => {
 p.innerText = `Texte numéro ${index + 1}`;
 p.style.border = "1px solid grey";
 p.style.padding = "8px";
});
```

Bonus : transformer un HTMLCollection en tableau

Si tu utilises **getElementsByClassName**, tu ne peux pas faire forEach() directement :

```
let elements = document.getElementsByClassName("maClasse");
// Erreur : elements.forEach is not a function
```

Solution : transformer en tableau JS classique :

```
let tabElements = Array.from(elements);
tabElements.forEach(el => {
 el.style.color = "blue";
});
```

Pièges fréquents à éviter

- Ne pas utiliser **innerHTML** avec du contenu utilisateur (risque d’injection XSS)
- Bien vérifier que l’élément **existe** avant de le modifier (sinon erreur JS)
- Ne pas oublier que ***getElementsBy…*** retourne une collection vivante (elle se met à jour si le DOM change), contrairement à querySelectorAll

NodeList vs HTMLCollection — Quelle est la différence ?

Comment on obtient une HTMLCollection ?

Une HTMLCollection est renvoyée par des méthodes DOM qui retournent des éléments HTML uniquement, souvent en direct (live).

```
// HTMLCollection
const elements = document.getElementsByClassName('ma-classe');
// HTMLCollection
const balises = document.getElementsByTagName('div');
```

Une HTMLCollection se met à jour automatiquement si le DOM change (c’est “live”).

NodeList vs HTMLCollection — Quelle est la différence ?

Comment on obtient une NodeList ? Une NodeList est renvoyée :

- par querySelectorAll() (le plus courant aujourd'hui)
- ou quand on utilise childNodes (inclut tous les types de nœuds, pas que les éléments HTML)

```
// NodeList
const items = document.querySelectorAll('.ma-classe');
// NodeList (texte, commentaires, éléments…)
const enfants = document.body.childNodes;
```

Une NodeList ne se met pas à jour automatiquement. Elle est statique (photographiée à l'instant).

NodeList vs HTMLCollection — Ce qu'ils ont en commun

Quoi utiliser ?

- Utilisez querySelectorAll() : plus moderne, plus flexible
- Si vous avez une NodeList et que vous voulez .map() ou .filter() :

```
// Conversion en vrai tableau
const array = Array.from(maNodeList);
```

HTMLCollection & NodeList : ce qu'ils ont en commun :

- Ce sont tous les deux des **collections d'éléments du DOM** (comme des tableaux).
- Ils sont **indexés** (maListe[0], maListe[1], etc.).
- On peut les parcourir avec une **boucle for classique**.
- On les obtient via certaines **méthodes DOM**.

NodeList vs HTMLCollection — Quoi utiliser ?

- **Utilisez querySelectorAll() :** plus moderne, plus flexible
- Si vous avez une NodeList et que vous voulez .map() ou .filter() :

```
// Conversion en vrai tableau
const array = Array.from(maNodeList);
```

NodeList vs HTMLCollection — Quelle est la différence ?

Aspect	HTMLCollection	NodeList
Type	Objet de type HTMLCollection	Objet de type NodeList
Méthodes	Pas de .forEach() nativement	Peut utiliser .forEach()
Dynamique ?	Oui — collection **vivante** (elle se met à jour si le DOM change)	Non — **statique**, figée au moment de la sélection
Obtenu avec	getElementsByClassName, getElementsByTagName	querySelectorAll ou childNodes

NodeList vs HTMLCollection : l'exemple concret

HTMLCollection : vivant !

```
<ul id="liste">
 <li>Un</li>
 <li>Deux</li>
</ul>
<script>
 let items = document.getElementsByTagName("li");
 console.log(items.length); // Affiche 2
 let ul = document.getElementById("liste");
 let nouveau = document.createElement("li");
 nouveau.innerText = "Trois";
 ul.appendChild(nouveau);

 console.log(items.length); // Affiche 3 : la collection s'est mise à jour automatiquement
</script>
```

NodeList vs HTMLCollection : l'exemple concret

NodeList : figée

```
let items = document.querySelectorAll("li");
console.log(items.length); // 2

let nouveau = document.createElement("li");
ul.appendChild(nouveau);

console.log(items.length); // 2 toujours ! Il faut refaire querySelectorAll
```

Ressources

NodeList : c'est quoi ?
https://developer.mozilla.org/fr/docs/Web/API/NodeList

HTMLCollection : c'est quoi ?
https://developer.mozilla.org/fr/docs/Web/API/HTMLCollection

FreeCodeCamp
https://www.freecodecamp.org/news/dom-manipulation-htmlcollection-vs-nodelist/

innerText
https://developer.mozilla.org/en-US/docs/Web/API/HTMLElement/innerText

HTMLStyleElement
https://developer.mozilla.org/fr/docs/Web/API/HTMLStyleElement