Manipulation du DOM en JavaScript

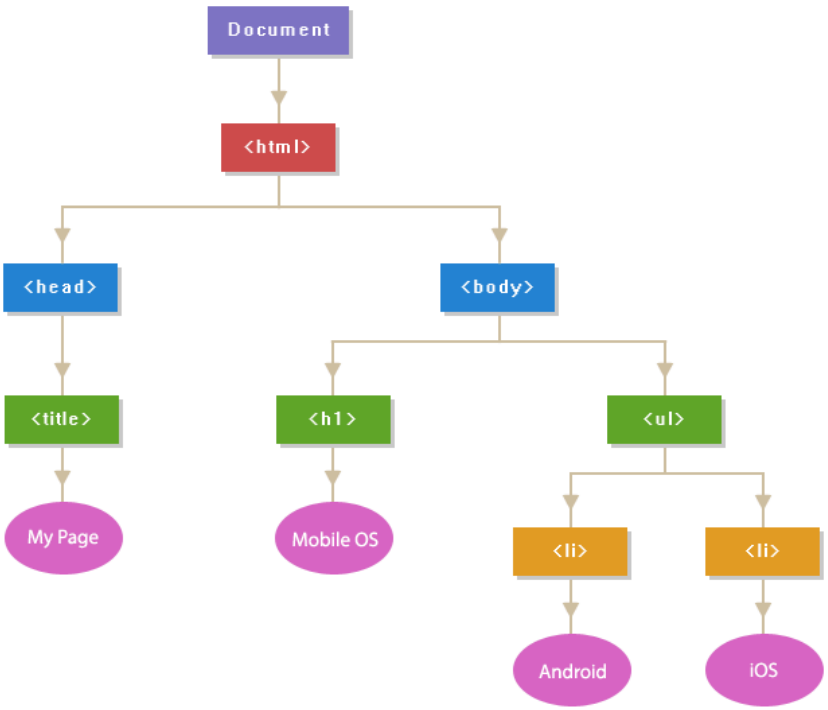
C’est quoi le DOM en JavaScript ?

DOM = Document Object Model

➡️ C’est une **représentation en mémoire** de la **structure HTML** d’une page web, sous forme **d’arbre**.

> Grâce au DOM, JavaScript peut **accéder**, **modifier** ou **supprimer** n’importe quel élément HTML dynamiquement.

Structure DOM



Exemple HTML

```html
<!DOCTYPE html>
<html>
  <head>
    <title>Ma page</title>
  </head>
  <body>
    <h1 id="titre">Bonjour Yasine</h1>
    <p class="text">Ceci est un paragraphe.</p>
    <button onclick="changerTitre()">Changer le titre</button>
  </body>
```

```
</html>
```

➡️ Cette structure HTML est représentée dans le navigateur sous forme d'un **arbre DOM**.

Vue simplifiée en arbre :

```
document
├── html
│   ├── head
│   │   └── title
│   └── body
│       ├── h1#titre
│       ├── p.text
│       └── button
```

Manipuler le DOM avec JavaScript

✅ 1. Accéder à un élément

🎯 Objectif

Avant de modifier un élément HTML (texte, style, ajout…), on doit d'abord **le sélectionner**.
JavaScript nous donne plusieurs **méthodes** pour accéder aux éléments du DOM.

📌 Méthodes principales pour accéder à un élément

Méthode	Description	Exemple
`getElementById()`	Sélectionne un **élément par son ID**	`document.getElementById("monId")`
`getElementsByClassName()`	Sélectionne tous les **éléments avec une classe**	`document.getElementsByClassName("maClasse")`
`getElementsByTagName()`	Sélectionne tous les **éléments avec une balise HTML**	`document.getElementsByTagName("p")`
`querySelector()`	Sélectionne le **premier élément** qui correspond à un **sélecteur CSS**	`document.querySelector(".maClasse")`
`querySelectorAll()`	Sélectionne **tous les éléments** qui correspondent à un **sélecteur CSS**	`document.querySelectorAll("div.rouge")`

A. `getElementById()`

Utilisation : quand tu veux accéder à un élément **unique** avec un `id`.

Exemple HTML

```html
<h1 id="titre">Bonjour</h1>
```

JavaScript

```javascript
let titre = document.getElementById("titre");
console.log(titre.textContent); // Affiche : Bonjour
```

B. `getElementsByClassName()`

Utilisation : retourne **une liste (HTMLCollection)** d'éléments avec la même classe.

Exemple HTML

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

JavaScript

```javascript
let elements = document.getElementsByClassName("info");
console.log(elements[0].textContent); // "Texte 1"
console.log(elements.length);         // 2
```

C. `getElementsByTagName()`

Utilisation : sélectionne tous les éléments d'un **même type** (balise).

```html
<div>Premier</div>
<div>Deuxième</div>
```

```javascript
let divs = document.getElementsByTagName("div");
console.log(divs[1].textContent); // "Deuxième"
```

D. `querySelector()`

Utilisation : plus **moderne**, on peut utiliser des sélecteurs CSS (`#id`, `.class`, `tag`)
⚠️ Retourne **le premier** élément trouvé !

```html
<p class="important">Paragraphe 1</p>
<p class="important">Paragraphe 2</p>
```

```javascript
let p = document.querySelector(".important");
console.log(p.textContent); // "Paragraphe 1"
```

E. `querySelectorAll()`

Utilisation : même principe que le précédent, mais il retourne **tous les éléments**.

```javascript
let tous = document.querySelectorAll(".important");
tous.forEach(p => console.log(p.textContent));
```

Récap visuel

```html
<div id="box" class="bloc">Hello</div>
```

```js
document.getElementById("box");       // Sélection par id
document.getElementsByClassName("bloc"); // Par classe
document.getElementsByTagName("div");    // Par balise
document.querySelector("#box");      // Par CSS
document.querySelectorAll(".bloc");  // Tous les .bloc
```

✅ 2. Modifier le contenu d'un élément HTML

🎯 Objectif :

Changer ce qu'il y a **à l'intérieur d'un élément HTML** (texte ou HTML) avec JavaScript.

📌 Les propriétés à connaître :

Propriété	Description
`textContent`	Change uniquement le **texte**
`innerHTML`	Change le **HTML** (balises incluses)
`innerText` *(optionnelle)*	Semblable à `textContent`, mais dépend plus du **rendu visuel** (rarement utilisée)

A. `textContent`

> Change ou récupère **le texte brut** (sans HTML)

🔹 Exemple HTML :

```html
<p id="paragraphe">Bonjour Yasine !</p>
```

🔹 JavaScript :

```js
let p = document.getElementById("paragraphe");
p.textContent = "Contenu modifié."; // ✅ change le texte
```

B. `innerHTML`

> Permet d'ajouter ou de modifier du **HTML** à l'intérieur d'un élément (ex : balises `<strong>`, `<em>`)

🔹 Exemple HTML :

```html
<p id="paragraphe">Bonjour</p>
```

🔹 JavaScript :

```js
let p = document.getElementById("paragraphe");
p.innerHTML = "<strong>Texte en gras</strong>"; // ✅ change le contenu + ajoute balise
```

Différence entre `textContent` et `innerHTML`

```js
element.textContent = "<b>Hello</b>";
// Affiche littéralement : <b>Hello</b>

element.innerHTML = "<b>Hello</b>";
// Affiche : Hello en gras (interprète le HTML)
```

Exemple complet

```html
<h1 id="titre">Titre original</h1>
<button onclick="changerTexte()">Changer le texte</button>

<script>
  function changerTexte() {
    let titre = document.getElementById("titre");
    titre.textContent = "Nouveau Titre 👋";
  }
</script>
```

Résumé

Propriété	Change le texte brut	Interprète le HTML
`textContent`	✅	❌
`innerHTML`	✅	✅

✅ 3. Modifier le style d'un élément HTML

🎯 Objectif :

Changer **l'apparence (couleur, taille, bordure, etc.)** d'un élément HTML avec **JavaScript**.

📌 Méthodes pour modifier le style

A. Modifier directement avec `.style`

Tu peux accéder à la propriété `style` d'un élément, puis définir des **propriétés CSS**.

✅ Syntaxe :

```js
element.style.nomDeLaPropriétéCSS = "valeur";
```

🔸 Exemple :

```html
<p id="texte">Texte stylé !</p>
```

```javascript
let p = document.getElementById("texte");
p.style.color = "red";            // couleur du texte
p.style.fontSize = "24px";        // taille de police
p.style.backgroundColor = "yellow"; // fond jaune
```

⚠️ Remarques :

- Les noms des propriétés CSS sont en **camelCase** en JavaScript.
  - Exemple : `background-color` devient `backgroundColor`
  - `font-size` → `fontSize`

B. Ajouter / enlever une classe CSS

C'est plus propre et plus flexible !

✅ Syntaxe :

```javascript
element.classList.add("maClasse");    // Ajoute une classe
element.classList.remove("maClasse"); // Retire une classe
element.classList.toggle("maClasse"); // Ajoute si absente, retire si présente
```

🔸 Exemple 1 (`classList.add()`) :

➕ **Ajoute** une ou plusieurs classes à un élément.

```html
<style>
  .rouge {
    color: red;
    font-weight: bold;
  }
</style>

<p id="paragraphe1">Texte 1</p>
<button onclick="ajouterClasse()">Ajouter classe</button>
```

```javascript
function ajouterClasse() {
  let p = document.getElementById("paragraphe1");
  p.classList.add("rouge");
}
```

📌 Résultat : Le texte devient rouge et gras.

🔸 Exemple 2 (`classList.remove()`) :

❌ **Supprime** une classe si elle existe.

```html
<style>
  .bleu {
    color: blue;
  }
```

```
</style>

<p id="paragraphe2" class="bleu">Texte 2</p>
<button onclick="supprimerClasse()">Supprimer classe</button>
```

```
function supprimerClasse() {
  let p = document.getElementById("paragraphe2");
  p.classList.remove("bleu");
}
```

📌 Résultat : Le texte redevient noir (style par défaut).

🔸 Exemple 3 (`classList.toggle()`) :

🔁 Ajoute la classe si elle est absente, la retire si elle est présente

```
<style>
  .surbrillance {
    background-color: yellow;
  }
</style>

<p id="paragraphe3">Texte 3</p>
<button onclick="basculerClasse()">Activer/Désactiver surbrillance</button>
```

```
function basculerClasse() {
  let p = document.getElementById("paragraphe3");
  p.classList.toggle("surbrillance");
}
```

📌 Résultat :

- Premier clic → ajoute fond jaune
- Deuxième clic → enlève le fond jaune
- Et ainsi de suite...
