



Révision JS : Le DOM

Webdev/CF2M - Naïm - 2024

Qu'est-ce que le DOM ?





Définition

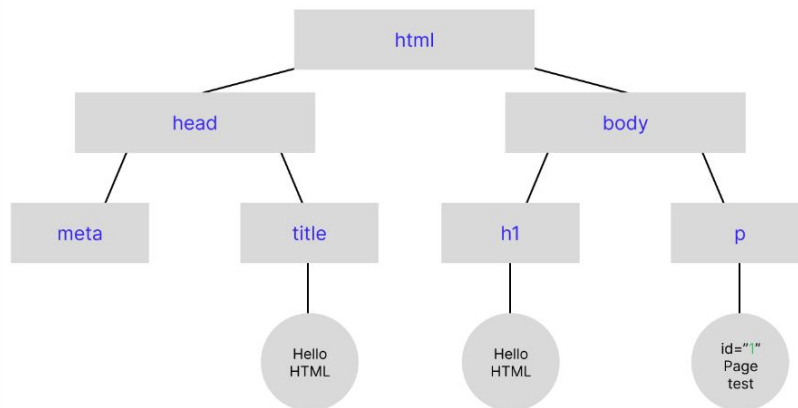
Le Document Object Model (DOM) est une **interface de programmation** pour les documents HTML et XML. Il fournit une page dont des programmes peuvent **modifier la structure, son style et son contenu**. Cette représentation du document permet de le voir comme un groupe structuré de nœuds et d'objets possédant différentes propriétés et méthodes. **Fondamentalement, il relie les pages Web aux scripts ou langages de programmation.**

Une page Web est un document. Celui-ci peut être affiché soit dans la fenêtre du navigateur, soit sous la forme de son code source HTML. Mais il s'agit du même document dans les deux cas. Le modèle objet de document proposé par le DOM fournit une autre manière de représenter, stocker et manipuler ce même document. Le DOM est une représentation entièrement orientée objet de la page Web, et peut être manipulé à l'aide d'un langage de script comme JavaScript.

(Source : *mdn web docs - Mozilla.org*)

Structure du DOM

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Hello HTML</title>
</head>
<body>
  <h1>
    Hello HTML
  </h1>
  <p id="1">
    Page test
  </p>
</body>
</html>
```



Le DOM représente la page web comme un “arbre”.

Les noeuds sont représentés par les rectangles , et le contenu par les ronds, les ronds peuvent également représenter des “méthodes” ou fonctions.



Pourquoi manipuler le DOM ?

- Ajouter, modifier ou supprimer des éléments HTML.
- Modifier le style CSS d'éléments spécifiques.
- Réagir aux événements utilisateurs (clics, saisies, etc.).

En résumé, c'est grâce aux manipulations du DOM qu'on peut rendre un site interactif (animation + envoi des requêtes correctes vers le back-end)



Manipuler le DOM

- Sélectionner des éléments :

```
<script>  
  const element = document.getElementById('1');  
  console.log(element);  
</script>
```

(Sélection de la balise avec l'id="1")

```
<script>  
  const element = document.getElementsByClassName('maClasse');  
  console.log(element);  
</script>
```


(Sélection de la classe avec le nom="maClasse")



```
<script>
  const element = document.querySelector('p');
  const allElements = document.querySelectorAll('p');
  const thirdElement = document.querySelector('p:nth-of-type(3)');
  console.log(element);
  console.log(allElements);
  console.log(thirdElement);
</script>
```

On peut également choisir de sélectionner des éléments selon leur <balisage> :

- Avec *querySelector* on peut sélectionner le premier élément de la page englobé par un <p>
- Avec *querySelector* combiné à *nth-of-type* on peut cibler l'élément selon sa position sur la page
- Avec *querySelector* combiné à *nth-child* on peut cibler l'élément selon sa position sur la page, quelle que soit la balise dans laquelle il est englobé (à partir de body)
- Avec *querySelectorAll* on peut sélectionner tous les éléments de la page englobé par des <p>

- 
- Avec *textContent* on peut modifier le texte au sein d'un élément qu'on a présélectionné. (ici avec la variable *element*).

```
element.textContent = 'Nouveau texte';
```

- Avec *innerHTML* on peut modifier le contenu HTML d'un élément.


```
thirdElement.innerHTML = '<strong>Texte en gras</strong>';
```

- *setAttribute* & *getAttribute* permettent de récupérer et de modifier les attributs des balises (id , classe , ...) , il faut évidemment préciser la “clé” afin de récupérer où d'envoyer les bonnes infos.

```
element.setAttribute('id', '2');  
title.getAttribute('class');
```

- *style* permet de modifier les propriétés CSS d'un élément

```
element.style.color = "red";
```


- 
- `createElement()` permet de créer un nouvel élément HTML.
On peut combiner `createElement()` avec d'autres méthodes
 - `remove()` permet de supprimer un élément html (à condition de le cibler correctement au préalable)

Gestion des évènements :

- `addEventListener()` ajoute un écouteur d'événement pour réagir à des actions de l'utilisateur.
- `removeEventListener()` permet de retirer l'écouteur ajouter avec la méthode ci-dessus.
- `event.preventDefault()` empêche le comportement par défaut d'un élément (empêcher un clic par exemple).
- `setTimeout()` , `setInterval()` permettent de gérer les événements dans le temps.