DOM : Présentation du DOM en JavaScript

Le Document Object Model, plus connu sous l'abréviation **DOM**, est un concept central en JavaScript lorsqu'on travaille sur des pages web.
Pensé comme une **représentation hiérarchique structurée**, il permet de manipuler et d'interagir avec les contenus HTML et XML.

Comprendre les fondamentaux du `DOM` est essentiel pour la **création dynamique de contenu** et le développement d'applications interactives.

1. Qu'est-ce que le DOM ?

Le `DOM` est la **représentation en mémoire d'une page web** interprétée par le navigateur.

Il s'agit d'une **interface de programmation** qui permet à JavaScript de manipuler dynamiquement le contenu, la structure et le style d'une page.
Avec le DOM, vous pouvez manipuler dynamiquement le contenu, la structure et le style d'une page.
En JavaScript, le DOM vous permet de :

- **Sélectionner** des éléments HTML.
- **Modifier** leurs propriétés.
- **Parcourir** la structure de la page.
- **Ajouter** ou **supprimer** des éléments dans le document.
- **Réagir** aux événements utilisateurs.

Apprendre à utiliser les méthodes du DOM est essentiel pour maîtriser les bases du développement web interactif.Pensé sous forme d'une **arborescence**, il structure les relations entre les différentes parties d'une page comme les balises HTML, leurs attributs, et leurs contenus.

1.1. Structure hiérarchique du DOM

Le `DOM` organise les éléments sous forme d'une **arborescence parent-enfant**.

Chaque élément HTML devient un **nœud** dans cette arborescence, permettant une **navigation facile** dans les éléments ainsi qu'une **manipulation simple**.

1.1.1. Syntaxe

Une des notions essentielles pour interagir avec le `DOM` est d'accéder à ses nœuds via des méthodes JavaScript.

Ces méthodes permettent de **sélectionner**, **naviguer**, **modifier** ou **créer** des éléments dans le `DOM`.

1.1.2. Exemple(s)

html

```html
<html>
  <head>
    <title>Exemple</title>
  </head>
  <body>
    <h1>Bienvenue</h1>
    <p>Ceci est un document HTML.</p>
  </body>
</html>
```

1.1.3. Représentation du DOM

Arborescence

```
html
├── head
│   └── title
│       └── "Exemple"
└── body
    ├── h1
    │   └── "Bienvenue"
    └── p
        └── "Ceci est un document HTML."
```

1.2. Rôle de JavaScript avec le DOM

> Le `DOM` est l'interface qui fait le lien entre votre script `JavaScript` et la structure de la page web.
>
> Il permet de **réagir aux actions utilisateur** et de rendre une page dynamique grâce à des modifications en temps réel.

1.2.1. Syntaxe

> Pour modifier un élément dans le `DOM`, une des opérations les plus courantes est l'actualisation du contenu texte ou des styles.

1.2.2. Exemple(s)

Exemple de sélection d'un élément et modification de son contenu :

javascript

```javascript
// Modification du texte d'un élément
const para = document.getElementById('monParagraphe');
para.textContent = 'Nouveau contenu';
```

1.3. Communication entre JavaScript et le DOM

> Le fonctionnement du `DOM` repose sur des API fournies par les navigateurs, appelées les `DOM APIs`.
>
> Ces APIs offrent une **interface standardisée** pour interagir avec les nœuds, les styles, et les événements liés à la page web.

2. Pourquoi apprendre le DOM ?

> Comprendre et maîtriser le DOM est essentiel pour tout développeur web. Cela facilite la création de pages interactives et l'implémentation d'éléments dynamiques en réponse à des actions d'utilisateur.
>
> Bien que son apprentissage nécessite de saisir des concepts comme la structure de l'arborescence ou les évènements, il offre une puissance inestimable pour le développement web.

💡 Conclusion

A retenir

- **Représentation hiérarchique** : Le DOM est une structure en arbre qui représente la page web, où chaque élément HTML est un nœud.
- **Interface de programmation** : Il permet à JavaScript d'accéder, de modifier et de manipuler les éléments de la page.
- **Dynamisme** : Grâce au DOM, on peut rendre une page interactive en modifiant son contenu ou ses styles en temps réel.
- **API standardisée** : Les navigateurs fournissent des méthodes comme `getElementById`, `createElement`, ou `appendChild` pour interagir avec le DOM.
- **Événements** : Le DOM permet de gérer les interactions utilisateur (clics, survols, etc.) via des gestionnaires d'événements.