Le DOM

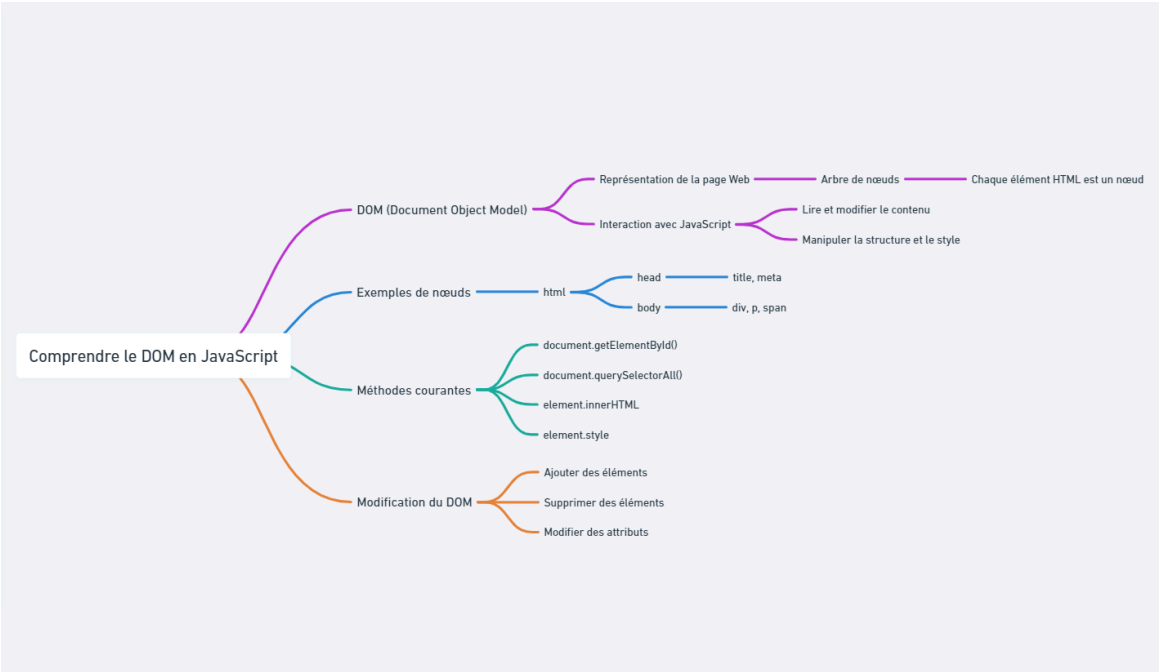
Qu'est-ce que le DOM?

Le DOM est une représentation orientée objet de la page web, qui permet aux langages de programmation comme JavaScript de modifier le contenu, la structure et le style des documents HTML. Pensez au DOM comme à un pont entre votre code JavaScript et la visualisation HTML/CSS de votre site.

```javascript
// Sélectionne l'élément avec l'ID 'ajouter' et ajoute un écouteur d'événements pour le clic.
document.getElementById('ajouter').addEventListener('click', function() {
    // Récupère la valeur du champ de texte avec l'ID 'maTache'.
    let tache = document.getElementById('maTache').value;

    // Vérifie si le champ de texte n'est pas vide.
    if (tache) {
        let li = document.createElement('li');

        // Ajoute le texte de la tâche à l'élément li.
        li.textContent = tache;

        // Ajoute l'élément li à la liste avec l'ID 'listeTaches'.
        document.getElementById('listeTaches').appendChild(li);

        // Réinitialise le champ de texte.
        document.getElementById('maTache').value = '';
    }
});
```

À la découverte du DOM

Le DOM (pour Document Object Model) est une représentation orientée objet de la page web, ce qui permet aux langages de programmation comme JavaScript d'interagir avec le contenu, la structure et le style de la page.



Javascript : Introduction au DOM

Chaque élément HTML est un nœud dans l'arbre du DOM. Par exemple, **<html>**, **<head>**, **<body>**, ** **, ****, etc., sont tous des nœuds.

Lire le DOM

En lisant le DOM, on obtient des informations sur les éléments, comme leurs attributs, leurs styles et leur contenu.

1. Sélection par Balise(getElementById, getElementsByTagName, getElementsByClassName)

Accéder aux Éléments :

- **document.getElementById('id')** : Sélectionne un élément par son ID.
- **document.getElementsByClassName('class')** : Sélectionne tous les éléments avec une classe spécifique.
- **document.getElementsByTagName('tag')** : Sélectionne les éléments par nom de balise.

```
// Sélection d'un élément par son ID
const elementById = document.getElementById('monElement');

// Sélection de tous les éléments avec la classe 'Conteneur'
const elementsByClassName = document.getElementsByClassName('Conteneur');

// Sélection de tous les éléments avec la balise 'li'
const elementsByTagName = document.getElementsByTagName('li');
```

2. Sélection par Sélecteur CSS (querySelector, querySelectorAll)

- **document.querySelector('selector')** : Sélectionne le premier élément qui correspond au sélecteur CSS.

```
// Code HTML avec une div ayant la classe "conteneur" et deux paragraphes à l'intérieur
<div class="conteneur">
<p>Paragraphe 1</p>
<p>Paragraphe 2</p>
</div>

// JavaScript : Sélectionne le premier paragraphe à l'intérieur d'une div avec la classe "conteneur"
const premierParagraphe = document.querySelector('.conteneur p');
```

- **document.querySelectorAll('selector')** : Sélectionne tous les éléments qui correspondent au sélecteur CSS.

```
// Code HTML avec une liste non ordonnée (ul) contenant deux éléments de liste (li)
<ul>
<li>Élément 1</li>
<li>Élément 2</li>
</ul>

// JavaScript : Sélectionne tous les éléments de liste (li) à l'intérieur de toutes les listes non ordonnées (ul)
const tousLesElementsLi = document.querySelectorAll('ul li');
```

`document.querySelectorAll` renvoie une **NodeList** (une collection d'éléments), même si un seul élément est trouvé.
Cette approche permet de cibler plusieurs éléments HTML à la fois pour les manipuler ou itérer à travers eux avec JavaScript.

Les sélecteurs du DOM en JavaScript sont des méthodes utilisées pour "sélectionner" des éléments HTML à partir de la page web. Ces sélections permettent ensuite de manipuler le contenu, le style, et le comportement des éléments.

Éditer le DOM

Éditer le DOM permet de modifier des éléments HTML, de changer des styles, d'ajouter ou de supprimer des éléments, etc. JavaScript permet de modifier le contenu (avec **innerHTML**, **textContent**), les attributs (avec **setAttribute**) et la structure (avec **appendChild**, **removeChild**).

1. Modification du Contenu(textContent, innerHTML)

La propriété **`element.textContent`** sert à changer ou récupérer le texte à l'intérieur d'un élément. Elle ne prend pas en compte les balises HTML, considérant tout le contenu comme du texte brut.

```
// Sélectionne l'élément du DOM avec l'ID 'maDiv'.
let moDiv = document.getElementById('maDiv');

// Modifie le contenu textuel de l'élément 'maDiv'.
monDiv.textContent = 'Nouveau Texte';
```

La propriété **`element.innerHTML`** permet de modifier ou de récupérer le contenu HTML d'un élément. Elle peut être utilisée pour insérer des balises HTML dans l'élément.

```
// Sélectionne l'élément du DOM avec l'ID 'maDiv'.
let maDiv = document.getElementById('maDiv');

// Modifie le contenu HTML de l'élément 'maDiv'.
maDiv.innerHTML = '<strong>Nouveau Texte</strong>';
```

2. *Modification de Style*

`element.style.property` : Permet de modifier le style CSS d'un élément. Les propriétés CSS en JavaScript sont généralement écrites en camelCase.

```
let maDiv = document.getElementById('maDiv');

// Modifie le background color de l'élément 'maDiv'.
maDiv.style.backgroundColor = 'lightblue';

// Modifie la taille de la police (font size) de l'élément 'maDiv'.
maDiv.style.fontSize = '20px';
```

3. *Ajout et Suppression d'Éléments*

Ajout d'Éléments :

- **`parent.appendChild(element)`** : Ajoute un nouvel élément comme dernier enfant de l'élément parent.

```
let nouvelleLi = document.createElement('li');
nouvelleLi.textContent = 'Nouvel élément';

// Sélectionne l'élément du DOM avec l'ID 'maListe' et ajoute le nouvel élément à la fin de la liste.
document.getElementById('maListe').appendChild(nouvelleLi);
```

Suppression d'Éléments :

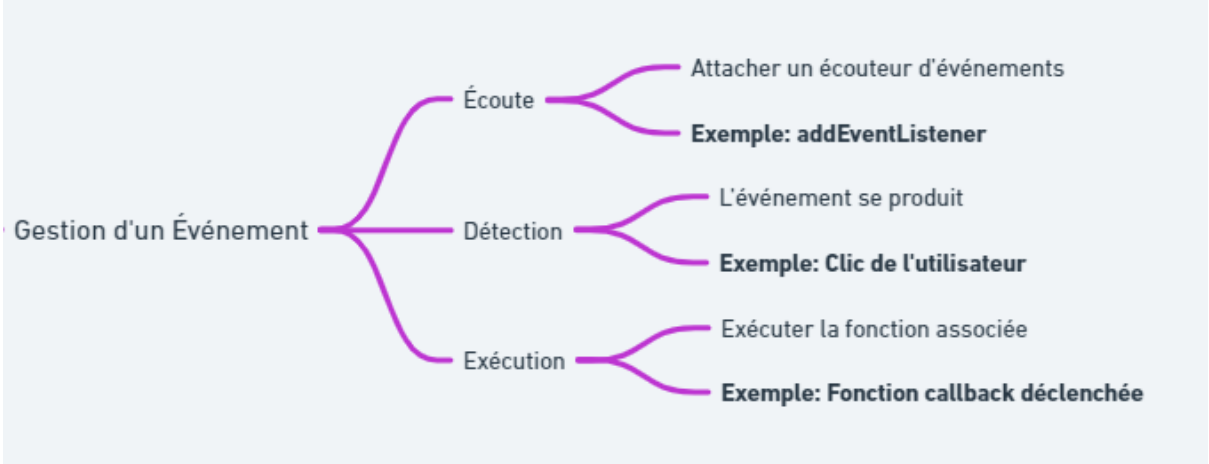
- **`parent.removeChild(element)`** : Supprime un élément enfant de l'élément parent.

Exemple :

```
let liste = document.getElementById('maListe');
let elementASupprimer = document.getElementById('elementASupprimer');

// Supprime l'élément spécifié de la liste.
liste.removeChild(elementASupprimer);
```

Les Événements

Les événements en JavaScript sont des actions ou occurrences qui se produisent dans le navigateur que vous pouvez utiliser pour déclencher des scripts.



Attacher un Écouteur d'Événement

La méthode **`addEventListener`** est utilisée pour attacher un écouteur d'événement à un élément.

```
// Attacher un écouteur d'événement 'click' à un bouton
let bouton = document.getElementById('monBouton');

bouton.addEventListener('click', function() {
    alert('Bouton cliqué!');
});
```

Syntaxe : **element.addEventListener(event, function, useCapture);**

Types d'événements

Les événements en JavaScript jouent un rôle crucial dans l'interaction des applications web. Ils rendent les sites web interactifs en permettant la gestion :

- de clics de souris
- d'entrées clavier
- de modifications de formulaires, et d'autres interactions.

Exemples d'évènements :

- **`load`** : Se déclenche lorsque la page ou une ressource est complètement chargée.
- **`click`** : Se déclenche lorsqu'un utilisateur clique sur un élément.
- **`dblclick`** : Se déclenche lorsqu'un utilisateur double-clique sur un élément.
- **`mouseover`** : Se déclenche lorsqu'un pointeur de souris passe sur un élément.
- **`submit`** : Se déclenche lors de la soumission d'un formulaire.

```
window.addEventListener('load', function() {
// Code à exécuter une fois que la page est entièrement chargée
});
```

Il y a encore beaucoup d'autres types d'événements à découvrir. Cela va des clics de souris basiques aux actions de toucher plus complexes, en passant par les interactions clavier et les changements dans les formulaires. Ces événements permettent de créer des expériences utilisateur intéressantes et dynamiques.

La Fonction de Rappel (Callback)

La fonction passée à `**addEventListener**` est appelée fonction de rappel (callback). Elle est exécutée lorsque l'événement se produit.

```
// Utilisation de l'objet event dans une fonction de rappel
bouton.addEventListener('click', function(event) {
    console.log('Élément cliqué:', event.target);
});
```

Cette fonction peut recevoir un objet `**event**` en argument, fournissant des détails sur l'événement survenu.