JavaScript - Le DOM

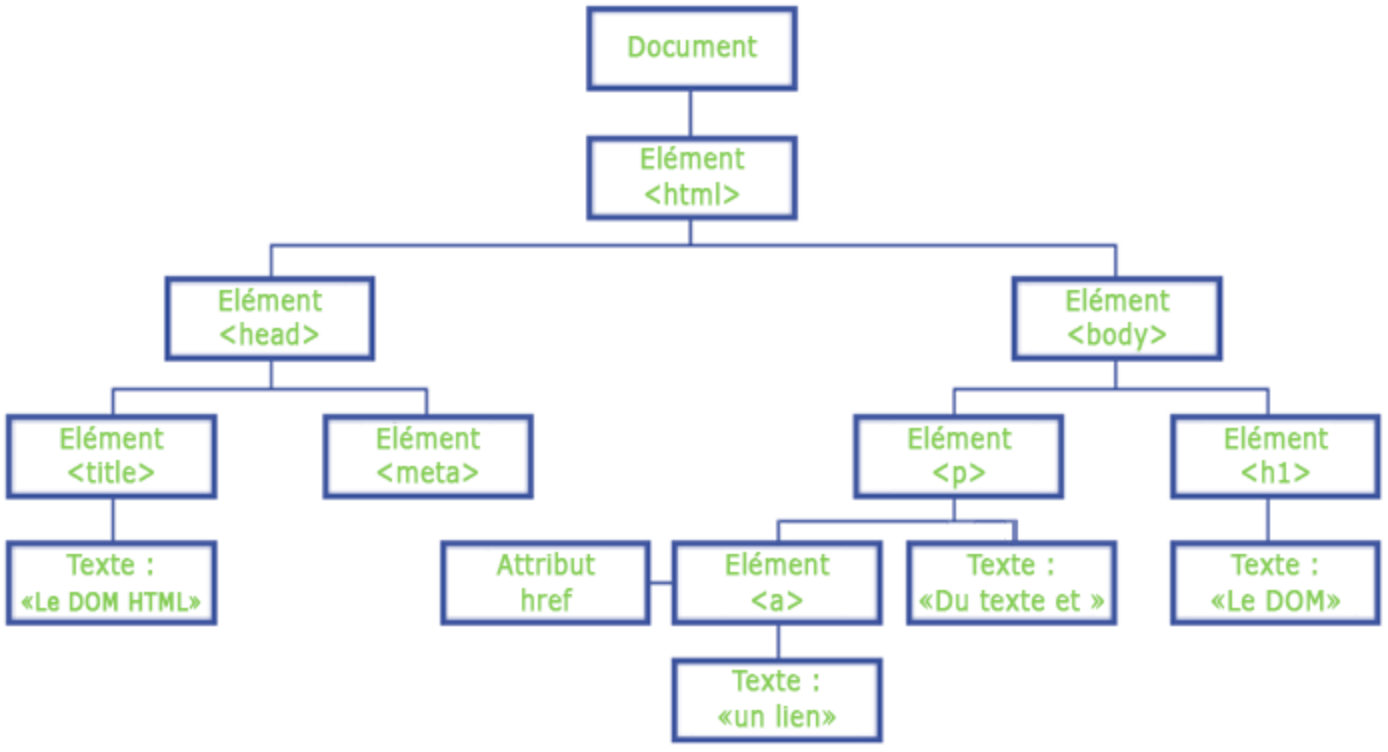
Qu’est-ce que le DOM?

> Le Document Object Model (DOM) est une interface de programmation normalisée par le W3C, qui permet à des scripts d'examiner et de modifier le contenu du navigateur web. Par le DOM, la composition d'un document HTML est représentée sous forme d'un jeu d'objets

Tout élément HTML sera un **objet en JavaScript avec ses propriétés et ses méthodes**.

Structure du DOM

Tout objet appartenant au DOM est un **node** (noeud).



Les méthodes de l'objet Document

- `getElementById("id")` : cible l'élément HTML avec l'id correspondant
- `getElementsByTagName("tag")` : retourne dans un tableau tous les éléments du même genre (ex: tous les `p`)
- `getElementsByClassName("cls")` : retourne dans un tableau tous les éléments d'une même classe
- `querySelector(".post")` : retourne le premier élément obtenu grâce à un sélecteur CSS
- `querySelectorAll(".post")` : retourne tous les éléments obtenus grâce à un sélecteur CSS

Le contenu d’un élément HTML

```html
<p id="hi">Salut <span>toi!</span></p>
```

```
let hi = document.getElementById("hi");
console.log(hi); // <p id="hi">Salut <span>toi!</span></p>

// All content in element
console.log(hi.innerHTML); // Salut <span>toi!</span>

// Only text content in element
console.log(hi.textContent); //Salut toi!

// Classes for this element
hi.className="me";
```

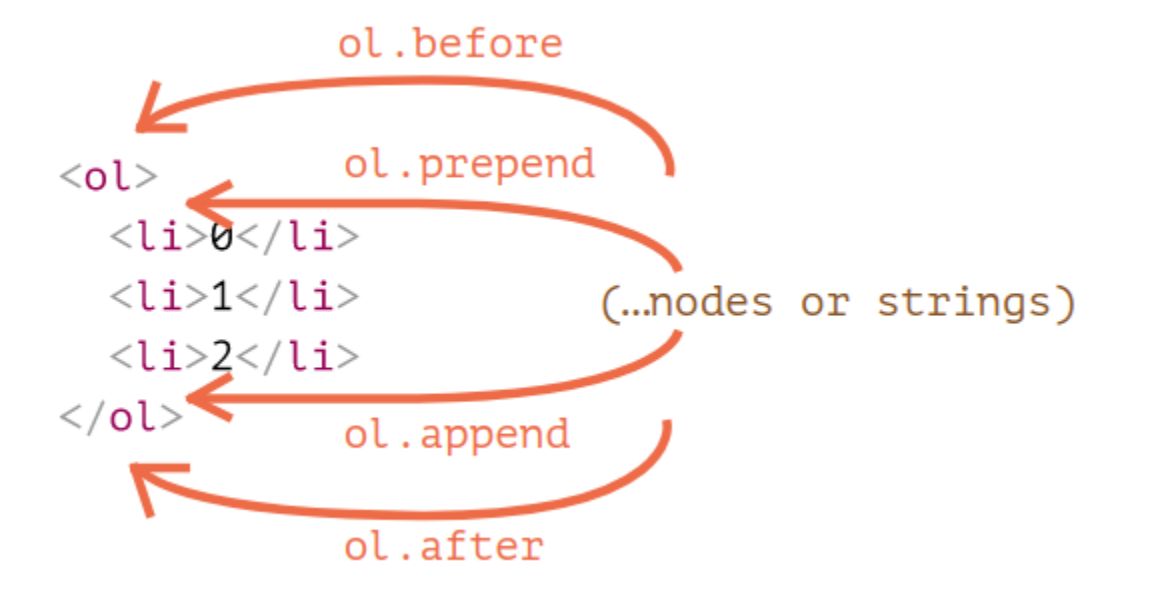
Création d'un objet HTML

- `document.createElement('p')` : Crée un nouvel élément **node** avec le tag correspndant. Ici, un `p`
- `document.createTextNode('text')` : Crée un nouvel élément **textNode** avec le texte correspondant. Ici, `text`

Méthodes d'insertion

- `node.append(el)` : insère **à la fin de node** l'élément el
- `node.prepend(el)` : insère **au début du node** l'élément el
- `node.before(el)` : Insère **avant le node** l'élément el
- `node.after(el)` : Insère **après le node** l'élément el
- `node.replaceWith(el)` : **Remplace le node** avec l'élément el

Schéma méthode d'insertion



Supprimer un node

`node.remove()` : supprime le node

Cloner un node

`elem.cloneNode(deep)` – Clone l'élément elem.
Si deep vaut `true`, le clone avec tous ses descendants

Style CSS et classes

N'utiliser la propriété `style` que lors de calculs.
Privilégier `classList` ou `className`

```
let marginTop=; /* complex calculations */
elem.style.marginTop = marginTop;

/* NB: Style properties are in camelCase */

// Replace whole classes for elem
elem.className = "alert alert-danger";

elem.classList.add("alert-disabled"); //alert alert-danger alert-disabled
```

Les méthodes de la propriété `classList`

- `elem.classList.add("class")` : Ajoute la classe.
- `elem.classList.remove("class")` : Supprime la classe
- `elem.classList.toggle("class")` : Ajoute la classe si elle n'est pas présente, sinon l'enlève
- `elem.classList.contains("class")` : Vérifie si la classe est présente, retourne true ou false

Gestion des événements

`elem.addEventListener(eventType,function)` :
eventType -> type d'événement:

- `click` : Au clic complet de la souris (appui et relachement du bouton)
- `mouseover` : Lorsque la souris survole l'élément
- `mouseout` : Lorsque la souris cesse de survoler l'élément
- `mousedown` : Lorsqu'on appuie sur le clic de la souris
- `mouseup` : Lorsqu'on relâche le clic de la souris

function -> la fonction à exécuter lors du déclenchement de l'événement

```
function changeText()
{
    // this is the current element
    this.textContent = "HIDDEN MESSAGE";
}

let p = document.querySelector(".text");
p.addEventListener("click",changeText);
```

Propagation des événements

- **phase de capture**: le parcours du DOM HTML en descendant
- **phase de bouillonnement** : la remontée du parcours HTML.
  C'est durant cette phase que les événements HTML se déclenchent.

Les événements de l'enfant s'exécuteront avant ceux du parent.

Si on désire que l'événement voulu s'exécute durant la phase de capture, il faut indiquer un troisième paramètre dans `addEventListener()` avec comme valeur `true`

Les propriétés de l'objet Event

- `target` : retourne l'élément qui a généré l'événement
- `currentTarget` : retourne l'élément qui est le gestionnaire de cet événement
- `type` : retourne le type de l'événement (ex: "click")
- `clientX` et `clientY` : la position en X et Y de l'endroit où l'utilisateur a émis l'événément
- `event.stopPropagation()` : Arrête la propagation des événements suite à l'appel de cette méthode
- `event.preventDefault()` : Stoppe l'événement