Parcourir le DOM

Comprendre comment parcourir l'arborescence du `DOM` est une compétence clé en développement web.

Le `DOM` étant organisé en structure hiérarchique, il est fondamental de savoir comment accéder aux éléments parents, enfants ou frères afin d'effectuer des manipulations précises.
Cela permet de modifier des parties spécifiques d'une page, d'interagir avec des éléments adjacents ou de naviguer dynamiquement dans toute la structure HTML.

---

1. Accéder aux nœuds parents

Les nœuds `parents` représentent l'élément directement supérieur à un nœud donné dans l'arborescence.

À l'aide des propriétés comme `parentNode` et `parentElement`, vous pouvez remonter dans la hiérarchie pour cibler l'élément contenant.

1.1. `parentNode`

La propriété `parentNode` retourne le nœud **parent immédiat** d'un élément, qu'il soit un nœud élément (`Element`) ou un autre type de nœud (texte, commentaire, etc.).

1.1.1. Syntaxe

javascript

```javascript
const parent = element.parentNode;
```

1.1.2. Exemple(s)

```html
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
</head>
<body>
    <header>
        <h1>Titre</h1>
    </header>
</body>
</html>
```

```javascript
const heading = document.querySelector("h1");
console.log(heading.parentNode); // affiche l'élément parent, peu importe son type (ici, le header)
```

1.2. parentElement

- Contrairement à `parentNode`, `parentElement` retourne uniquement un **nœud élément**.
- Si le parent est autre qu'un élément (par exemple, un nœud texte), cette propriété retourne `null`.

1.2.1. Syntaxe

```javascript
const parent = element.parentElement;
```

1.2.2. Exemple(s)

```javascript
const html = document.documentElement;
console.log(html.parentElement); // Retourne null, car <html> est la racine du document
```

---

2. Accéder aux nœuds enfants

> - Les nœuds `enfants` sont les éléments directement contenus dans un nœud.
> - Le DOM offre plusieurs propriétés pour accéder à ces enfants, qu'ils soient des nœuds élément ou d'autres types de nœuds.

2.1. `childNodes`

> Cette propriété retourne une **liste de tous les enfants** d'un nœud, y compris les nœuds texte et commentaires.

2.1.1. Syntaxe

```javascript
const enfants = element.childNodes;
```

2.1.2. Exemple(s)

```javascript
const list = document.querySelector("ul");
console.log(list.childNodes); // Affiche tous les nœuds enfants, y compris les textes
```

2.2. `children`

> La propriété `children` retourne uniquement les **nœuds éléments** enfants d'un nœud, en ignorant le texte et les commentaires.
>
> C'est une alternative plus pratique à `childNodes` lorsque vous souhaitez accéder uniquement aux éléments.
>
> Le retour est une collection `HTMLCollection`, qui ressemble à un tableau mais n'en est pas un.

2.2.1. Syntaxe

```javascript
const enfants = element.children;
```

2.2.2. Exemple(s)

```html
<body>
    <ul>
        <!-- commentaire -->
        <li>Item 1</li>
        <li>Item 2</li>
    </ul>
</body>
```

```javascript
const list = document.querySelector("ul");
for (const enfant of list.children) {
    console.log(enfant);
}
// <!-- commentaire -->
// <li>Item 1</li>
// <li>Item 2</li>
```

2.3. firstChild et lastChild

> Ces propriétés permettent d'accéder respectivement au **premier** et au **dernier nœud enfant**, quels que soient leurs types (texte, élément, etc.).

2.3.1. Syntaxe

```javascript
const premier = element.firstChild;
const dernier = element.lastChild;
```

2.3.2. Exemple(s)

```javascript
const list = document.querySelector("ul");
console.log(list.firstChild); // Affiche le premier enfant, peu importe son type : <!-- commentaire -->
console.log(list.lastChild); // Affiche le dernier enfant : <li>Item 2</li>
```

2.4. `firstElementChild` et `lastElementChild`

> Ces propriétés retournent respectivement le **premier** et le **dernier enfant** qui sont des nœuds élément.
> Elles ignorent les nœuds texte et commentaires.
> Ces propriétés sont particulièrement utiles pour accéder aux éléments enfants **sans se soucier des autres types de nœuds**.

2.4.1. Syntaxe

```javascript
const premier = element.firstElementChild;
const dernier = element.lastElementChild;
```

2.4.2. Exemple(s)

```html
<body>
    <ul>
        <li>Item 1</li>
        <li>Item 2</li>
        <li>Item 3</li>
        <li>Item 4</li>
    </ul>
</body>
```

```javascript
const list = document.querySelector("ul");
console.log(list.firstElementChild); // <li>Item 1</li>
console.log(list.lastElementChild); // <li>Item 4</li>
```

3. Accéder aux nœuds frères

> Les nœuds `frères` désignent les nœuds situés immédiatement avant ou après un nœud donné dans l'arborescence.
>
> Les propriétés comme `nextSibling` ou `previousSibling` permettent de naviguer entre les frères, qu'ils soient des éléments ou d'autres types de nœuds.

3.1. `nextSibling` et `previousSibling`

> Ces propriétés retournent respectivement le **nœud suivant** ou le **nœud précédent** dans l'arborescence, peu importe son type (texte, commentaire, élément...).

3.1.1. Syntaxe

javascript
```javascript
const suivant = element.nextSibling;
const precedent = element.previousSibling;
```

3.1.2. Exemple(s)

javascript
```javascript
const item = document.querySelector("li");
console.log(item.nextSibling); // Affiche le frère suivant, peu importe son type
console.log(item.previousSibling); // Affiche le frère précédent
```

3.2. `nextElementSibling` et `previousElementSibling`

> Ces propriétés retournent respectivement le **frère suivant** ou le **frère précédent**, uniquement s'ils sont des nœuds élément.
> Elles ignorent les nœuds texte et commentaires, ce qui les rend **plus pratiques** pour naviguer entre les éléments.
> Ces propriétés sont particulièrement utiles pour accéder aux éléments frères **sans se soucier des autres types de nœuds**.

3.2.1. Syntaxe

javascript
```javascript
const suivant = element.nextElementSibling;
const precedent = element.previousElementSibling;
```

3.2.2. Exemple(s)

```javascript
const item = document.querySelector("li");
console.log(item.nextElementSibling); // Le prochain li
console.log(item.previousElementSibling); // Le li précédent
```

💡 Conclusion

A retenir

- Traverser le `DOM` est essentiel pour accéder et manipuler les différents nœuds d'une page web.
- **Pour remonter dans la hiérarchie**, utilisez `parentNode` ou `parentElement` .
- **Pour accéder aux enfants**, privilégiez `childNodes` (pour tous les types de nœuds) ou `children` (uniquement les éléments).
- **Pour passer d'un frère à un autre**, utilisez `nextSibling` / `previousSibling` (tous types) ou `nextElementSibling` / `previousElementSibling` (uniquement éléments).
- **Pour accéder au premier/dernier enfant**, utilisez `firstChild` / `lastChild` (tous types) ou `firstElementChild` / `lastElementChild` (uniquement éléments).
- Il est tout de même recommandé d'utiliser les propriétés les plus spécifiques (`parentElement` , `children` , `nextElementSibling` , `firstElementChild` , etc.) pour éviter les nœuds texte et commentaires.