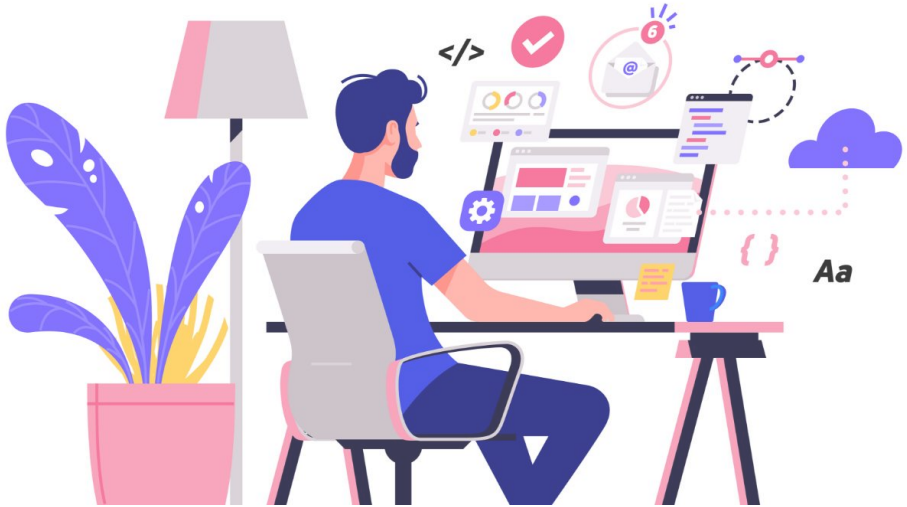
Javascript



Objectifs

- Sélectionner directement un noeud du DOM
  - Utiliser **getElementById()**
  - Utiliser **getElementsByClassName()**
  - Utiliser **getElementsByTagName()**
  - Utiliser **querySelector()** et **querySelectorAll()**

- Sélectionner un noeud du DOM depuis un autre noeud :
  - Utiliser **parentElement**
  - Utiliser **firstElementChild, lastElementChild**
  - Utiliser **nextElementSibling, previousElementSibling**

Le DOM est un arbre qui possède de nombreux noeuds.

Comment sélectionner précisément l’un de ses noeuds ? On peut le faire de plusieurs manières

getElementById()

document.getElementById() : Sélectionne un élément HTML par son **id**

- La méthode retourne un **Element** dont la valeur de l'attribut id est identique à celle passée en paramètres (*string*)
- Si aucun élément n'est trouvé, la méthode renvoie **null**

```
<p id="product"></p>
```

```
const myElement = document.getElementById('product');
```

```
    <p id="product"></p>
>
```

Après avoir sélectionné un élément, on peut agir dessus : ajouter du style, manipuler ses attributs, récupérer les parents ou les enfants.

C’est ce que l’on va voir par la suite.

Car nous, pour le moment, nous sommes bloqués : chaque élément HTML ne possède pas nécessairement d’id.

→ Voyons les autres méthodes pour sélectionner nos éléments.

getElementsByClassName()

document.getElementsByClassName() : Sélectionne un élément HTML par sa **class**

- La méthode retourne un **HTMLCollection** **(un tableau)** qui contient tous les enfants partageant une même **classe**
- Si aucun élément n'est trouvé, la méthode renvoie **null**

```
<section class="container">
    <!-- some content -->
</section>
<!-- Some content -->
<section class="container">
    <!-- some content -->
</section>
```

```
const containerList = document.getElementsByClassName('container');
console.log(containerList);
```

getElementsByClassName()

```
▼HTMLCollection(2) i
  ▶0: section.container
  ▶1: section.container
    length: 2
```

On récupère par conséquent **un tableau d'éléments.**

Il ne nous reste plus qu'à chercher à l'intérieur du tableau via l'index 🤗

```
console.log(containerList[0]);
```

```
▶<section class="container">…</section>
```

getElementsByTagName()

document.getElementsByClassName() : Sélectionne un élément HTML par sa **balise**

- Comme pour la méthode **getElementsByTagName()** retourne un **HTMLCollection** **(un tableau)**
- Cette fois-ci, c'est un tableau qui contient tous les enfants partageant une même **balise**
- Si aucun élément n'est trouvé, la méthode renvoie **null**

```
const sectionList = document.getElementsByTagName('section');
console.log(sectionList);
```

```
<section class="container">
    <!-- some content -->
</section>
<!-- Some content -->
<section class="container">
    <!-- some content -->
</section>
```

```
▼HTMLCollection(2) [sect
  ▶0: section.container
  ▶1: section.container
   length: 2
```

querySelector()

- **document.querySelector() :** Sélectionne un élément HTML par son **id**, sa **classe** ou sa **balise**
- Cette méthode **résume les 3 méthodes précédentes** ! C'est génial 🙌
- L'écriture de l'argument est proche du CSS

⚠️ Si c'est une **classe** ou une **balise**, la méthode querySelector() renvoie la **première occurrence**

```
const myForm = document.querySelector('#form');
const firstContainerWhichMatches = document.querySelector('.container');
const firstSectionWhichMatches = document.querySelector('section');
```

📝 À noter : il est possible de cibler très précisément un élément (comme on le ferait en CSS avec plusieurs sélecteurs) :

```
const p = document.querySelector('#hero-content p')
```

querySelectorAll()

Si c'est une classe ou une balise, la méthode **querySelectorAll() renvoie une HTMLCollection (tableau)**

```
const allSectionsWhichMatches = document.querySelectorAll('section'); // toutes les occurences
```

Récupérer précisément un élément du DOM est très utile.

Ça nous permet également de récupérer ses parents/enfants/frères

parentElement

element.parentElement : Sélectionne l'élément HTML **parent**

```
<section class="container">
    <div>
        <p>Lorem ipsum dolor
    </div>
```

```
const myDiv = document.querySelector('div');
console.log(myDiv.parentElement);
```

```
▶<section class="container">…</section>
```

firstElementChild

element.firstElementChild : Sélectionne le premier élément HTML **enfant**

```
<section class="container">
    <div>
        <p>Lorem ipsum dolor
    </div>
```

```
const myContainer = document.querySelector('.container');
console.log(myContainer.firstElementChild);
```

```
▶<div>…</div>
```

lastElementChild

element.lastElementChild : Sélectionne le dernier élément HTML **enfant**

```
<section class="container">
    <div>
        <p>Lorem ipsum dolor
    </div>
    <div>
        <p>Lorem ipsum dolor
    </div>
    <div>
        <p>Lorem ipsum dolor
    </div>
```

```
const myContainer = document.querySelector('.container');
console.log(myContainer.lastElementChild);
```

```
▶ <div>…</div>
```

nextElementSibling

element.nextElementSibling : Sélectionne le prochain **frère direct**

```
<ul id="menu">
    <li>Home</li>
    <li>Products</li>
    <li class="current">C
    <li>Careers</li>
    <li>Investors</li>
    <li>News</li>
    <li>About Us</li>
</ul>
```

```
let current = document.querySelector('.current');
let nextSibling = current.nextElementSibling;
console.log(nextSibling);
```

```
▼<li>
    ::marker
    "Careers"
  </li>
```

previousElementSibling

element.previousElementSibling : Sélectionne le précédent **frère direct**

```
<ul id="menu">
    <li>Home</li>
    <li>Products</li>
    <li class="current">C
    <li>Careers</li>
    <li>Investors</li>
    <li>News</li>
    <li>About Us</li>
</ul>
```

```
let current = document.querySelector('.current');
let prevSiblings = current.previousElementSibling;
console.log(prevSiblings);
```

▶ li

Pratiquons !