Javascript

Objectifs

- Créer un élément
- Ajouter un élément au **DOM**
- Modifier un élément
- Supprimer un élément du **DOM**

Créer un élément

- **document.createElement(element) :** création d'un élément

```
const div = document.createElement('div'); // création de l'élément
```

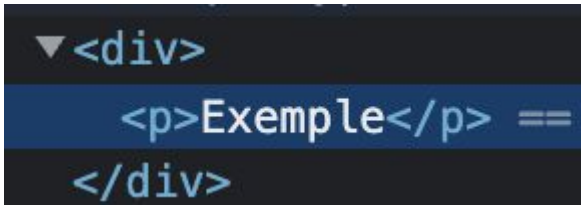
Je viens de créer **le squelette de mon élément.**

C'est bien mais ça ne suffit pas : il faut écrire son **contenu**

Écrire le contenu d'un élément (HTML)

- **element.innerHTML=** "code HTML"

```
div.innerHTML = '<p>Exemple</p>';
```



JS

Écrire le contenu d'un élément (Texte uniquement)

- **element.innerText =** (“chaîne de caractères”)

```
const p = document.querySelector('#hero-content p')
p.innerText = "Changement de texte"
```

```
<div id="hero-content" class=
  <p>Changement de texte</p>
</div>
```

innerText vs innerHTML

D’un point de vue performances, **innerText** est bien meilleur. Mais c’est normal : il ne permet d’écrire que le texte d’un élément.

innerHTML permet quant-à-lui d’écrire le HTML (et donc aussi le texte) d’un élément. Il est donc bien plus utile et plus bien souple.

L’utilisation de l’un ou de l’autre dépendra donc du cas de figure rencontré 🤓

JS

Ajout d'un id/class sur un élément

- **div.id = "my-id" :** ajout d'un id
- **div.classList.add**("my-class") : ajouter une classe
- **div.classList.remove**("my-class") : supprimer une classe
- **div.classList.toggle**("my-class") : ajouter/supprimer une classe

```
div.id = "hero-content"; // ajout d'un id
div.classList.add("container");  // ajout d'une classe
div.classList.remove("container"); // suppression d'une classe
// Si la classe est présente, la supprimer. Si elle est absente, l'ajouter
div.classList.toggle("container");
```

```
▼<div id="hero-content" class="container">
   <p>Exemple</p>
 </div>
```

Ajouter un élément au DOM

- **parent.appendChild(enfant)** : ajout d'un élément dans le **DOM**, en tant qu'enfant d'un parent précisé

```
document.body.appendChild(div); // ajout de la div dans le body
```

L'élément viendra **s'ajouter à la fin** de mon élément parent.

C'est très utile !

Mais appendChild() possède en réalité **2 effets**

Zoom sur appendChild

- **Effet 1 :** ajouter un enfant à l'élément sélectionné si l'enfant n'est pas dans le DOM

```
const menu = document.querySelector('#menu');
const pChild = document.createElement('li');
pChild.innerText = "Ceci est l'enfant créé"
menu.appendChild(pChild);
```

- Home
- Products
- Support personnalisé
- Careers
- Investors
- News
- About Us
- Ceci est l'enfant créé

Zoom sur appendChild

- **Effet 2 :** déplacer l'élément s'il existe déjà dans le DOM

```
const menu2 = document.querySelector('#menu2');
menu2.appendChild(pChild);
```

- Home
- Products
- Support personnalisé
- Careers
- Investors
- News
- About Us

- Ceci est l'enfant créé

```
▶<ul id="menu">…</ul>
▼<ul id="menu2"> == $0
  ▶<li>…</li>
  </ul>
```

JS

Supprimer un élément

- **parent.removeChild**(enfant)

```
const menu2 = document.querySelector('#menu2');
menu2.removeChild(menu2.firstElementChild);
```

```
<ul id="menu2">
    <li>Home</li>
    <li>Products</li>
    <li>About Us</li>
</ul>
```

```
▼<ul id="menu2">
  ▶<li>…</li> == $
  ▶<li>…</li>
</ul>
```

- Products
- About Us

Pratiquons !