Semaine 10 - Modifier dynamiquement le DOM

Avec JavaScript, il existe plusieurs API pour créer ou supprimer des éléments HTML du DOM (Document, Node, ParentNode et Element). Cette note de cours **se concentre** sur les propriétés et méthodes liées aux **APIs Document** (qui représente le point d'entrée pour accéder au contenu de la page Web qui est formée de l'arbre du DOM) et **Element** (qui représente les éléments d'un document HTML).

Avec JavaScript, l'ajout d'un élément HTML se fait en trois temps:

1. **Création** de l'élément HTML;
2. **Affectation** de **style**(s) ou de **classe CSS**;
3. **Insertion dans le document,** et ce n'est qu'à ce moment-là qu'il sera véritablement « ajouté » et visible.

Création d'un élément HTML

La création d'un élément se fait avec la méthode **createElement()** du document. La syntaxe est la suivante:

let elementHTML = document.createElement(tagName);

où :

- **tagName** est une chaîne spécifiant le type d'élément ou de **balise** HTML à créer.

Par exemple :

```
let nouvelleDiv = document.createElement("div");
```

Dans cet exemple, on crée un nouvel élément ** **. Cet élément est créé, **mais n'est pas inséré dans le document**, il n'est donc pas encore visible. Cela dit, on peut déjà travailler dessus, en ajoutant des styles ou des classes CSS, ou même des événements.

Affectation de style(s) CSS

Pour affecter des styles CSS à l'élément HTML créé, on peut, comme vu précédemment, utiliser sa propriété **«style»,** ou ajouter une **classe CSS** par programmation avec l'**API classList**. Exemple :

Ajout de styles CSS - exemple

```
let nouvelleDiv = document.createElement("div");
nouvelleDiv.style.width= "20vw";
nouvelleDiv.style.height= "10vh";
nouvelleDiv.style.border= "2px solid black";
```

Ajout d'une classe CSS - exemple

```
let nouvelleDiv = document.createElement("div");
nouvelleDiv.classList.add("la-classe-CSS");
```

Insertion dans le document

Pour ajouter l'élément HTML créé dans la page, trois principales méthodes peuvent être utilisées : **append(), before()** et **after()**.

Méthode append()

La méthode append() permet d'ajouter un ou plusieurs éléments HTML à la fin de la liste des enfants d'un **nœud parent** spécifié. La syntaxe est la suivante:

elementParent.append(enfant); ou elementParent.append(enfant1, enfant2, etc.)

où:

- **elementParent** est le nœud (ou balise HTML) où l'on souhaite insérer l'élément HTML créé.
- **enfant** est le nœud à ajouter sous l'élément parent.

Par exemple :

```
let nouvelleDiv = document.createElement("div");
nouvelleDiv.classList.add("la-classe");

let laSection= document.querySelector("section");
laSection.append(nouvelleDiv);
```

Dans cet exemple, on crée un nouvel élément ** **. Cet élément est inséré au sein de la balise **<section>** présente dans le document.

AVANT	APRÈS

`<body>`
`  <section></section>`
`</body>`
	`<body>`
	`  <section>`
	`  <div class="la-classe"></div>`
	`  </section>`
	`</body>`

Méthode before()

La méthode **before()** permet d'ajouter un ou plusieurs éléments HTML juste avant un élément de référence. La syntaxe est la suivante.

elementReference.before(nouvelElement) ou elementReference.before(nouvelElement1, nouvelElement2, etc.)

où:

- **elementReference** est l'élément HTML de référence ;
- **nouvelElement** est l'élément HTML à ajouter avant l'élément de référence.

Par exemple:

```
let nouvelleDiv = document.createElement("div");
nouvelleDiv.classList.add("la-classe");

let laSection = document.querySelector("section");
laSection.before(nouvelleDiv)
```

Dans cet exemple, on crée un nouvel élément . Cet élément est inséré avant la balise <section> présente dans le document.

AVANT	APRÈS

`<body>`
`  <section></section>`
`</body>`
	`<body>`
	`  <div class="la-classe"></div>`
	`  <section></section>`
	`</body>`

Méthode after()

La méthode **after()** permet d'ajouter un ou plusieurs éléments HTML juste après un élément de référence. La syntaxe est la suivante:

elementReference.after(nouvelElement) ou elementReference.after(nouvelElement1, nouvelElement2, etc.)

où:

- **elementReference** est l'élément HTML de référence;
- **nouvelElement** est l'élément HTML à ajouter après l'élément de référence.

```
let nouvelleDiv = document.createElement("div");
nouvelleDiv.classList.add("la-classe");

let laSection = document.querySelector("section");
laSection.after(nouvelleDiv)
```

Dans cet exemple, on crée un nouvel élément . Cet élément est inséré avant la balise <section> présente dans le document.

AVANT	APRÈS

`<body>`
`  <section></section>`
`</body>`
	`<body>`
	`  <section></section>`
	`  <div class="la-classe"></div>`
	`</body>`

Vérifier qu'un élément HTML contient des nœuds enfants

Pour vérifier si un élément HTML contient ou non des nœuds enfants, on peut utiliser la propriété : **elementHTML.children** combinée avec la valeur **length** : **elementHTML.children**.**lenght**. La propriété **elementHTML.children** renvoie un **tableau** indiquant les nœuds enfants de cet élément, la propriété **elementHTML.children**.**lenght** indiquera quant à elle le **nombre** d'enfants de cet élément HTML Si la valeur est plus grande que 0 (>0), l'élément parent contient donc des enfant…

Par exemple :

```
let nouvelleDiv = document.createElement("div");
let laSection = document.querySelector("section");
laSection.appendChild(nouvelleDiv);

console.log(laSection.children.length > 0);//Retourne : true, si il y a des enfants
```

Supprimer un élément HTML

Pour **supprimer** un élément HTML, on utilise la méthode **remove()** directement sur l'élément à supprimer.

La syntaxe est donc la suivante:

elementHTML.remove();

où:

- **elementHTML** est l'élément du DOM ou de la structure HTML à supprimer.

Par exemple :

```
let laDiv = document.querySelector("div");
LaDiv.remove();
```

Dans cet exemple, on supprime l'élément ** ** inclus dans la page HTML :

AVANT	APRÈS

`<body>`
`  <section>`
`  <div class="la-classe"></div>`
`  </section>`
`</body>`
	`<body>`
	`  <section></section>`
	`</body>`

Supprimer le premier enfant d'un élément HTML avec firstElementChild

Pour supprimer le premier enfant d'un élément HTML donné, on peut utiliser la propriété **firstElementChild** de ce conteneur parent. Dans cet exemple, on supprime le **premier** élément ** ** inclus dans la balise <section> :

```
let laSection = document.querySelector("section");
let premierElementEnfant = laSection.firstElementChild;
premierElementEnfant.remove();
```

Dans cet exemple, on supprime le **premier** élément ** ** inclus dans la balise <section>.

AVANT	APRÈS

<body>
  <section>
   1
   2
   3
  </section>
</body>
	<body>
	  <section>
	   2
	   3
	  </section>
	</body>

Supprimer le dernier enfant d'un élément HTML avec lastElementChild

Pour supprimer le dernier enfant d'un élément HTML donné, on peut utiliser la propriété **lastElementChild**. Par exemple :

```
let laSection=document.querySelector("section");
let dernierElementEnfant =  laSection.lastElementChild;
dernierElementEnfant.remove();
```

Dans cet exemple, on supprime le **dernier** élément ** ** inclus dans la balise <section>.

AVANT	APRÈS

<body>
  <section>
   1
   2
   3
  </section>
</body>
	<body>
	  <section>
	   1
	   2
	  </section>
	</body>

À explorer...

L'API Element contient de nombreuses autres propriétés et méthodes, comme *prepend()*, *insertAdjacentElement()*, *replaceWidth()*, etc… On invite donc les étudiants curieux et intéressés à explorer davantage cette API par eux-mêmes...

Quelques références utiles sur les APIs Document et Element

Pour une vue complète des possibilités de ces APIs, vous pouvez consulter les références suivantes:

- Modzilla (Novembre 2022). Document- Référence Web API | MDN. Repéré à: https://developer.mozilla.org/fr/docs/Web/API/Document
- Modzilla (Septembre 2022). Element- Référence Web API | MDN. Repéré à: https://developer.mozilla.org/fr/docs/Web/API/Element
- Modzilla (Septembre 2022). Manipuler des documents. repéré à: https://developer.mozilla.org/fr/docs/Learn/JavaScript/Client-side_web_APIs/Manipulating_documents
- OpenClassrooms. (Mars 2023). Modifiez le DOM - Écrivez du JavaScript pour le web. Repéré à https://openclassrooms.com/fr/courses/5543061-ecrivez-du-javascript-pour-le-web/5577491-modifiez-le-dom