Bachelor CDWFS
Applications web dynamiques et interactives
2025-2026

Javascript

HTML Dynamique

Pré-requis
Connaître le balisage html
Être capable d'insérer du code javascript dans une balise script ou dans un fichier JS
Savoir utiliser les outils de développement
Connaître la méthode document.getElementById()

Code de démonstration
- ciblage.html

Ajouter & Supprimer des éléments

Ajout d'éléments

L'ajout d'élément HTML se fait en 2 étapes :

1e étape : créer un élémént virtuel

La fonction **document.createElement** permet de créer un élément. Cette méthode prend en paramètre le nom de la balise de notre élément et renvoie l'élément créé.
Exemple :

```
const newElt = document.createElement('div');
```

Un élément créé avec cette fonction ne fait pas encore partie du document, vous ne le verrez donc pas sur votre page. Pour le voir, il va d'abord falloir l'ajouter en tant qu'enfant à un élément.

2e étape : ajouter l'élément sur le document html

Pour ajouter un élément après d'autres éléments, on peut utiliser la méthode **append**.
Voici un exemple :

```
const newElt = document.createElement('div');
let elt = document.getElementById('main');
elt.append(newElt);
```

append est une méthode qui s'applique sur l'élément parent (ici : **elt**)

elt.append(element) prend en paramètre l'élément à ajouter en tant qu'enfant.

Autres méthodes :

node.**append**(...nodes or strings) – insère dans node, à la fin,

node.**prepend**(...nodes or strings) – insère dans node, au début,

node.**before**(...nodes or strings) –- insère juste avant node,

node.**after**(...nodes or strings) –- insère juste après node,

Travaux dirigés : Exercices 3 et 4

Suppression d'éléments

Quelques exemples

```
const newElt = document.createElement('div');
let elt = document.getElementById('main');
elt.appendChild(newElt);
elt.remove(newElt); // Supprime l'élément newElt de l'élément elt
elt.replaceWith(document.createElement('article'), newElt); // Remplace l'élément newElt par
un nouvel élément de type article
```

Modifier les textes

Lecture ...

On peut récupérer le contenu d'un élément avec la propriété innerHTML ou textContent. Par exemple :

```
<body>
      <h1 id='titre'>Titre <span>principal</span></h1>
      <p>Un paragraphe</p>
</body>
```

On peut récupérer le contenu du titre avec le code JavaScript suivant :

```
const titre = document.getElementById('titre') ;
const contenu = titre.innerHTML ; // Retourne « Titre <span>principal</span> »
const contenu = titre.textContent ; // Retourne « Titre principal »
```

… et écriture

Par exemple, avec le code JavaScript suivant :

```
let elt = document.getElementById('main');
elt.textContent = "Une ligne de texte";
```

l'élément qui a l'id 'main' aura un nouveau contenu ; le HTML deviendra donc :

```
<div id="main">Une ligne de texte</div>
```

Pour aller plus loin : innerHTML

Par exemple, avec le code JavaScript suivant :

```
let elt = document.getElementById('main');
elt.innerHTML = '<ul><li>Elément 1</li><li>Elément 2</li></ul>';
```

l'élément qui a l'id 'main' aura un nouveau contenu ; le HTML deviendra donc :

```
<div id='main'>
    <ul>
        <li>Elément 1</li>
        <li>Elément 2</li>
    </ul>
</div>
```

Travaux dirigés : Exercices 1

Valeurs des inputs de formulaire

Pour récupérer les valeurs des balises input des formulaires, on utilise la propriété **value**.

Ex

```
<form>
      <input type='text' name='login'/>
      <button type='submit'>Envoyer</button>
</form>
```

On peut récupérer la valeur du champ saisi par l'internaute avec le code JavaScript suivant :

```
const login = document.querySelector('input').value ;
```

Modifier les styles

Ajout / Suppression de classes

Il est aussi possible d'accéder directement à la liste des classes d'un élément avec la propriété **classList** .

Exemple :

Partie html

```
<div class="header top-box green element">
```

Partie Js :

```
const elt = document.querySelector('div') ;
```

Voici quelques méthodes possibles :

```
elt.classList.add('nouvelleClasse');// Ajoute la classe nouvelleClasse à l'élément
elt.classList.remove('nouvelleClasse'); // Supprime la classe nouvelleClasse
elt.classList.contains('nouvelleClasse');   // Retourne false
elt.classList.replace('green', 'blue'): // Remplace green par blue
```

Travaux dirigés : Exercices 2

Modification des propriétés de style

Avec la propriété **style** , vous pouvez récupérer et modifier les différents styles d'un élément.

Pour modifier la couleur d'un élément :

```
elt.style.color = '#fff';      // Change la couleur du texte de l'élément à blanche
```

Pour modifier la couleur d'arrière-plan d'un élément :

```
elt.style.backgroundColor = '#000';
```

Pour modifier la graisse d'un élément :

```
elt.style.fontWeight = 'bold'; // Met le texte de l'élément en gras
```

Les propriétés utilisables correspondent aux propriété css,
en adoptant l'écriture « CamelCase »

CSS	JS
background-color	backgroundColor
font-weight	fontWeight

Pour aller plus loin

Ajouter supprimer des attributs

Pour définir ou remplacer les attributs d'un élément, vous pouvez utiliser la fonction **setAttribute**.
element.setAttribute(name, value) prend en paramètres le nom de l'attribut et sa valeur et ne retourne rien.
Vous pouvez utiliser les fonctions **getAttribute** et **removeAttribute** pour avoir encore plus de contrôle sur les attributs.
Voici quelques exemples avec elt faisant référence à un élément de type input :

```
elt.setAttribute('type', 'password');   // Change le type de l'input en un type password
elt.setAttribute('name', 'my-password');    // Change le nom de l'input en my-password
elt.getAttribute('name');               // Retourne my-password
```

L’identifiant

On dispose du code html suivant :

```
<body>
      <h1 id='titre'>Titre <span>principal</span></h1>
      <p>Un paragraphe</p>
</body>
```

On peut récupérer l’id d’un élément avec la propriété id. Par exemple :

```
const titre = document.getElementById('titre') ;
const identifiant = titre.id; // Retourne « titre »
```