# Méthodologie d'ajout d'un `addEventListener`

## 1. Comment s'y prendre ?

La méthode `addEventListener` est une fonctionnalité essentielle en JavaScript pour rendre vos pages interactives.

Elle permet d'attacher des gestionnaires d'événements à des éléments HTML, comme réagir à un clic, une saisie ou une soumission de formulaire.

Voici une méthodologie simple et efficace en 5 étapes pour l'utiliser correctement.

### 1.1. Identifier l'élément cible

Avant d'ajouter un événement, vous devez d'abord sélectionner l'élément HTML auquel vous souhaitez attacher cet événement.

Cela se fait généralement à l'aide de méthodes comme `querySelector`, `getElementById`, ou `getElementsByClassName`.

#### 1.1.1. Pourquoi ?

L'élément cible est celui qui déclenchera l'événement (par exemple, un bouton, un champ de formulaire, etc.).

#### 1.1.2. Exemple :

```javascript
// Sélectionner un bouton avec un ID
const bouton = document.querySelector('#monBouton');

// Sélectionner un champ de formulaire avec une classe
const champTexte = document.querySelector('.monChamp');
```

### 1.2. Déterminer le type d'événement

Un événement est une action utilisateur ou du navigateur (comme un clic, une saisie, ou un survol).

Vous devez spécifier quel type d'événement vous souhaitez écouter.

#### 1.2.1. Pourquoi ?

Chaque type d'événement correspond à une action spécifique. Par exemple :

- `click` : Lorsqu'un utilisateur clique sur un élément.
- `input` : Lorsqu'un utilisateur saisit ou modifie un champ.
- `submit` : Lorsqu'un formulaire est soumis.

#### 1.2.2. Exemple :

```javascript
// Type d'événement pour un clic
const typeEvenement = 'click';

// Type d'événement pour une saisie
const typeEvenementSaisie = 'input';
```

## 1.3. Créer une fonction gestionnaire (callback)

> La fonction gestionnaire (callback) est le code qui sera exécuté lorsque l'événement se déclenche.
>
> Elle peut être :

- Nommée : Définie séparément pour être réutilisée.
- Anonyme : Définie directement dans `addEventListener` si elle est simple.

#### 1.3.1. Pourquoi ?

La fonction gestionnaire contient la logique que vous souhaitez exécuter en réponse à l'événement (par exemple, afficher un message, valider un formulaire, etc.).

#### 1.3.2. Exemple de fonction nommée :

```javascript
function afficherMessage() {
    console.log('Bouton cliqué !');
}
```

#### 1.3.3. Exemple de fonction anonyme :

```javascript
const afficherMessage = function() {
    console.log('Bouton cliqué !');
};
```

## 1.4. Ajouter l'écouteur d'événement

> Une fois l'élément cible, le type d'événement, et la fonction gestionnaire définis, utilisez la méthode `addEventListener` pour attacher l'événement.

#### 1.4.1. Pourquoi ?

`addEventListener` permet de lier un événement à un élément HTML. Vous pouvez également ajouter plusieurs événements au même élément sans les écraser.

#### 1.4.2. Exemple :

```javascript
// Ajouter un événement "click" au bouton
bouton.addEventListener('click', afficherMessage);

// Ajouter un événement "input" au champ de texte
champTexte.addEventListener('input', function() {
    console.log('Texte modifié :', champTexte.value);
});
```

---

## 1.5. Tester et ajuster

> Une fois l'écouteur ajouté, testez votre code pour vérifier que l'événement est bien capturé et que la fonction gestionnaire fonctionne comme prévu.
>
> Si nécessaire, ajustez la logique de votre fonction ou corrigez les erreurs.

#### 1.5.1. Pourquoi ?

Tester garantit que votre code fonctionne comme attendu et que l'événement est bien déclenché.

#### 1.5.2. Exemple :

```javascript
// Tester si le clic sur le bouton affiche le message
bouton.addEventListener('click', function() {
    console.log('Test réussi : le bouton a été cliqué.');
});
```

---

## Exemple complet

> Voici un exemple complet qui suit cette méthodologie :

```javascript
// 1. Identifier l'élément cible
const bouton = document.querySelector('#monBouton');
const champTexte = document.querySelector('#monChamp');

// 2. Déterminer le type d'événement
const typeEvenementClic = 'click';
const typeEvenementSaisie = 'input';

// 3. Créer une fonction gestionnaire
function afficherMessage() {
    console.log('Bouton cliqué !');
}

function afficherTexte(event) {
    console.log('Texte saisi :', event.target.value);
}

// 4. Ajouter l'écouteur d'événement
bouton.addEventListener(typeEvenementClic, afficherMessage);
champTexte.addEventListener(typeEvenementSaisie, afficherTexte);

// 5. Tester et ajuster
console.log('Les écouteurs d\'événements ont été ajoutés avec succès.');
```

---

## Bonnes pratiques

1. Vérifiez que l'élément existe avant d'ajouter un événement :

```javascript
const bouton = document.querySelector('#monBouton');
if (bouton) {
    bouton.addEventListener('click', afficherMessage);
}
```

2. Utilisez des fonctions nommées pour les gestionnaires complexes :
   Cela améliore la lisibilité et facilite la suppression des événements avec `removeEventListener`.
3. Nettoyez les écouteurs inutiles :
   Supprimez les écouteurs d'événements lorsque vous n'en avez plus besoin pour éviter les fuites de mémoire :

```javascript
bouton.removeEventListener('click', afficherMessage);
```

4. Utilisez les options avancées si nécessaire :
   Par exemple, pour exécuter un événement une seule fois :

```javascript
bouton.addEventListener('click', afficherMessage, { once: true });
```