

# TP – JavaScript : Gestion d'Événements et Interactions Utilisateur

## — Introduction : Comprendre JavaScript

Qu'est-ce que JavaScript ?

JavaScript est un langage de programmation créé en 1995 par Brendan Eich alors qu'il travaillait chez Netscape. Son objectif initial était de donner vie aux pages web statiques, en permettant des interactions dynamiques : imaginez des menus qui s'ouvrent, des formulaires qui valident vos données en temps réel, des animations fluides, et même des mini-jeux directement dans votre navigateur !

💡 Attention : Ne confondez jamais JavaScript avec le langage Java ! Malgré une partie de leur nom en commun, ce sont deux langages de programmation distincts, créés par des entités différentes et pour des usages très variés. JavaScript est principalement un langage côté client (exécuté par le navigateur), tandis que Java est un langage généraliste souvent utilisé côté serveur ou pour des applications lourdes.

## Objectifs pédagogiques

Ce TP est une **initiation** à la programmation avec JavaScript dans le contexte du **développement web**.

À travers 4 mini-jeux/outils, vous allez :

- Apprendre à **écouter les événements utilisateur** ( `click` , `input` , etc.)
- Manipuler le **DOM (Document Object Model)** avec JS
- Interagir avec des éléments HTML via JavaScript (création, suppression, mise à jour)
- Gérer des **timers** ( `setInterval` , `setTimeout` )
- Dynamiser vos pages web

---

## Structure du projet

Créez un dossier contenant **trois fichiers distincts** :

- `TP-VotreNom.html` → structure HTML (squelette de la page)
- `TP-VotreNom.css` → styles CSS (couleurs, formes, mise en page)
- `JSVotreNom.js` → logique JavaScript (comportement, interactions)

⚠ La **séparation HTML/CSS/JS** est **obligatoire** dans ce TP.


---

## Partie 1 – Jeu 1 : Clique sur la cible


## Objectif

Ce jeu met en œuvre la création dynamique d'éléments HTML, la gestion des événements utilisateur, et la manipulation du DOM.

L'objectif est de faire apparaître des cibles de manière aléatoire, à cliquer le plus rapidement possible pour marquer des points.

 Ce que vous allez apprendre :

- Générer des éléments dynamiquement ( `createElement` )
- Positionner un élément avec `style.left` et `style.top`
- Réagir aux clics utilisateur ( `addEventListener` )
- Utiliser `setInterval` et `setTimeout` pour gérer l'apparition et la disparition d'éléments

 Question : Pourquoi utilise-t-on `clearInterval()` à chaque démarrage du jeu ?

---

## Étapes

### ◆ HTML

```
<section>
  <h2>Jeu 1 : Clique sur la cible</h2>
  <p>Score : <span id="score">0</span></p>
  <button id="startTarget">Démarrer</button>
  <button id="stopTarget">Arrêter</button>
  <div id="targetZone"></div>
</section>
```

### ◆ CSS

- `.target` : cercle rouge positionné de manière absolue
- `#targetZone` : zone grise avec `position: relative`

### ◆ JavaScript

- Créez une fonction qui, toutes les 1,5 secondes :
    - Ajoute une cible `.target` à une position aléatoire
    - Supprime la cible après 1 seconde
  - Lorsqu'on clique sur la cible :
    - Elle disparaît
    - Le score augmente
- 

## Partie 2 – Jeu 2 : Mini Quiz

## Objectif

Ce mini-quiz introduit la création dynamique de formulaires et la vérification de saisie utilisateur.

🎯 Ce que vous allez apprendre :

- Générer du contenu HTML via JavaScript
- Lire et comparer les réponses utilisateur
- Gérer l'état d'un formulaire (désactiver champs/boutons)
- Structurer des données sous forme de tableau d'objets

💡 Question : Pourquoi utilise-t-on `toLowerCase()` avant de comparer les réponses ?

---

## 🔧 Étapes

### ♦ HTML

```
<section>
  <h2>Jeu 2 : Mini Quiz</h2>
  <div id="quiz"></div>
  <p>Score quiz : <span id="quizScore">0</span></p>
</section>
```

### ♦ JavaScript

- Stockez les questions comme objets avec une propriété `q` (question) et `r` (réponse)
  - Pour chaque question :
    - Créez un champ de saisie
    - Un bouton "Valider"
    - Comparez la réponse saisie à la bonne réponse
    - Mettez à jour le score et désactivez les champs corrects
- 

## 🎨 Partie 3 – Outil : Changer la couleur

### 🎯 Objectif

Apprendre à manipuler les **styles CSS via JavaScript** en temps réel.

🎯 Ce que vous allez apprendre :

- Utiliser un champ `input type="color"`
- Écouter l'événement `input` pour une mise à jour immédiate
- Modifier une propriété CSS ( `style.backgroundColor` ) dynamiquement

💡 Question : Quelle est la différence entre `input` et `change` pour un champ de couleur ?

---

## 🔧 Étapes

### ♦ HTML

```
<section>
  <h2>Jeu 3 : Change la couleur</h2>
  <input type="color" id="colorPicker">
  <div id="colorBox">Zone colorée</div>
</section>
```


### ◆ JavaScript

- Récupérez le color picker et la boîte à colorer
- Ajoutez un `addEventListener("input", ...)`
- Appliquez la valeur du picker à `box.style.backgroundColor`


## Partie 4 – Chronomètre

### Objectif

Créer un chronomètre simple avec démarrage, pause et réinitialisation.

 Ce que vous allez apprendre :

- Gérer un compteur dans le temps ( `setInterval` )
- Mettre à jour dynamiquement le contenu d'un élément ( `textContent` )
- Contrôler un processus avec des boutons ( `start` , `pause` , `reset` )

 Question : Pourquoi faut-il utiliser `clearInterval()` avant de redémarrer le chronomètre ?

### Étapes

#### ◆ HTML

```
<section>
  <h2>Jeu 4 : Chronomètre</h2>
  <p id="chrono">0 s</p>
  <button id="startChrono">Démarrer</button>
  <button id="pauseChrono">Pause</button>
  <button id="resetChrono">Réinitialiser</button>
</section>
```

#### ◆ JavaScript

- Lancez un intervalle ( `setInterval` ) qui incrémente une variable `chrono`
- Mettez à jour le contenu de `#chrono` chaque seconde
- Ajoutez les fonctions pause et réinitialisation

### Vérifications finales

- ☐ Fichiers HTML, CSS et JS sont bien séparés

- ☐ Chaque jeu fonctionne indépendamment
  - ☐ Les fonctions sont bien commentées
  - ☐ Les événements sont gérés avec `addEventListener`, pas dans le HTML
- 



## Pour aller plus loin (facultatif)

- Ajouter un **mode sombre** avec un bouton pour basculer les couleurs
- Enregistrer le **score maximum** dans le navigateur avec `localStorage`
- Ajouter une **limite de temps** ou un compte à rebours

🎉 Bravo ! Vous avez mis en œuvre les bases essentielles de l'interactivité web avec JavaScript.