TD3 - JavaScript : Interactivité de CinéScore

Contexte pédagogique

Après avoir construit la structure HTML (TD1) et la mise en forme CSS (TD2) de CinéScore, vous allez maintenant ajouter de l'interactivité à votre site. Vous créerez un fichier **script.js** unique lié à toutes vos pages HTML. Ce TD couvre les notions essentielles de JavaScript côté client : sélection du DOM, événements, manipulation dynamique, validation de formulaires, stockage local et appels API.

Arborescence

```
cinescore/
  index.html
  catalogue.html
  film_viceVersa2.html
  ajout.html
  styles.css
  script.js
  images/
    viceVersa2.jpg
```

Liaison du script

Ajoutez la balise suivante juste avant la fermeture `</body>` dans **toutes** vos pages HTML :

```
<script src="script.js"></script>
```

Exercice 1 - Sélection du DOM et premiers événements

Dans cet exercice, vous apprendrez à sélectionner des éléments HTML depuis JavaScript et à réagir aux actions de l'utilisateur.

Rappels utiles

```
// Sélection d'éléments
document.getElementById('monId')          // par id
document.querySelector('.maClasse')       // premier élément correspondant
document.querySelectorAll('p')           // tous les éléments correspondants

// Écouter un événement
element.addEventListener('click', function() {
  // code à exécuter au clic
});

// Modifier le contenu et le style
element.textContent = 'Nouveau texte';
element.innerHTML = '<strong>HTML</strong>';
element.style.color = 'red';
element.classList.add('maClasse');
element.classList.toggle('visible');
```

1.1 - Message de bienvenue dynamique (index.html)

Consignes :

- Sélectionnez le paragraphe de la section `.intro` et modifiez son contenu en fonction de l'heure du jour :
  - Avant 12h : « Bon matin ! Découvrez les films du moment. »
  - Entre 12h et 18h : « Bon après-midi ! Envie d'une séance cinéma ? »
  - Après 18h : « Bonsoir ! C'est l'heure idéale pour un bon film. »
- Utilisez `new Date()` et la méthode `.getHours()` pour obtenir l'heure actuelle.

Indice : const heure = new Date().getHours();

1.2 - Navigation active

Consignes :

- Au chargement de chaque page, récupérez le nom du fichier courant avec `window.location.pathname`.
- Parcourez tous les liens du <nav> avec `querySelectorAll('nav a')`.
- Ajoutez la classe CSS `active` au lien dont le `href` correspond à la page courante.
- Dans votre CSS, ajoutez un style pour `nav a.active` (par exemple : `background-color: rgba(255,255,255,0.3); border-radius: 5px;`).

CSS à ajouter dans styles.css :

```
nav a.active {
  background-color: rgba(255, 255, 255, 0.3);
  border-radius: 5px;
}
```

Indice : Utilisez element.getAttribute('href') pour comparer avec le nom du fichier courant. La méthode .includes() peut vous aider.

1.3 - Effet au clic sur les film-cards (index.html)

Consignes :

- Sélectionnez toutes les cartes de films (`.film-card`) avec `querySelectorAll`.
- Pour chaque carte, ajoutez un écouteur d'événement `click`.
- Au clic, basculez (`toggle`) une classe `film-selected` sur la carte.
- Dans votre CSS, définissez `.film-selected` avec une bordure colorée (par exemple `border: 3px solid #e63946`).
- Affichez dans la console (`console.log`) le titre du film cliqué.

CSS à ajouter dans styles.css :

```
.film-selected {
  border: 3px solid #e63946;
}
```

Indice : Pour récupérer le titre, utilisez card.querySelector('h3').textContent

Exercice 2 - Manipulation dynamique du DOM

Vous allez apprendre à créer, modifier et supprimer des éléments HTML dynamiquement.

Rappels utiles

```
// Créer un élément
const div = document.createElement('div');
div.className = 'ma-classe';
div.textContent = 'Contenu';

// Ajouter au DOM
parent.appendChild(div);
parent.insertBefore(div, referenceElement);

// Supprimer un élément
element.remove();
```

2.1 - Compteur de films sélectionnés (index.html)

Consignes :

- Créez dynamiquement (en JS, pas en HTML) un élément <p> avec l'id `compteur` et insérez-le juste avant la grille de films `.films-grid`.
- Son contenu initial : « 0 film(s) sélectionné(s) ».
- À chaque clic sur une carte (exercice 1.3), mettez à jour ce compteur en comptant les éléments ayant la classe `.film-selected`.

Indice : document.querySelectorAll('.film-selected').length vous donne le nombre de films sélectionnés.

2.2 - Système de recherche / filtre pour le catalogue (catalogue.html)

Consignes :

- Ajoutez dans votre HTML (dans la section du catalogue, avant le tableau) un champ de recherche :

HTML à ajouter dans catalogue.html (avant le <table>) :

```
<input type="text" id="recherche" placeholder="Rechercher un film...">
```

CSS à ajouter dans styles.css :

```
#recherche {
  width: 100%;
  padding: 10px;
  border: 2px solid #ddd;
  border-radius: 5px;
  font-size: 1em;
  margin-bottom: 20px;
  transition: border-color 0.3s ease;
}

#recherche:focus {
  outline: none;
  border-color: #457b9d;
  box-shadow: 0 0 5px rgba(69, 123, 157, 0.3);
}
```

- En JavaScript, écoutez l'événement `input` sur ce champ.

• À chaque frappe, récupérez la valeur saisie (en minuscules avec `.toLowerCase()`).

• Parcourez toutes les lignes du `<tbody>` du tableau.

• Pour chaque ligne, récupérez le texte complet (`row.textContent.toLowerCase()`).

• Si le texte contient la valeur recherchée, affichez la ligne (`row.style.display = ''`), sinon cachez-la (`row.style.display = 'none'`).

Indice : textContent.toLowerCase().includes(valeur) renvoie true ou false.

2.3 - Bouton « Retour en haut »

Consignes (sur toutes les pages) :

• Créez dynamiquement un bouton `<button>` avec l'id `btn-top` et le texte « ↑ Haut ».

• Ajoutez-le au `document.body`.

• Stylisez-le en JS : position fixed, en bas à droite, initialement caché (`display: none`).

• Écoutez l'événement `scroll` sur `window` : si `window.scrollY > 300`, affichez le bouton ; sinon, cachez-le.

• Au clic sur le bouton, faites défiler la page vers le haut avec `window.scrollTo({ top: 0, behavior: 'smooth' })`.

Exercice 3 - Validation de formulaire (ajout.html)

La validation côté client améliore l'expérience utilisateur en vérifiant les données avant l'envoi. Vous allez ajouter une validation JavaScript au formulaire d'ajout de film.

Rappels utiles

```
// Récupérer la valeur d'un champ
const valeur = document.getElementById('monChamp').value;

// Empêcher l'envoi du formulaire
form.addEventListener('submit', function(event) {
  event.preventDefault(); // empêche le rechargement
  // ... validation ...
});

// Vérifier qu'un champ n'est pas vide
if (valeur.trim() === '') { /* champ vide */ }
```

3.1 - Validation à la soumission

Consignes :

- Sélectionnez le formulaire (`document.querySelector('form')`).
- Écoutez l'événement `submit` et appelez `event.preventDefault()`.
- Vérifiez les règles suivantes :
  - Le titre doit contenir au moins 2 caractères.
  - La synopsis doit contenir au moins 10 caractères.
  - Un genre doit être sélectionné (valeur non vide).
- Si une règle n'est pas respectée, affichez un message d'erreur en rouge sous le champ concerné.
- Si tout est valide, affichez une alerte : « Film ajouté avec succès ! ».

3.2 - Affichage des erreurs

Consignes :

- Créez une fonction `afficherErreur(champId, message)` qui :
  - Supprime l'erreur précédente s'il y en a une (pour éviter les doublons).
  - Crée un `<span>` avec la classe `erreur`, le texte du message, et la couleur rouge.
  - L'insère après le champ concerné avec `insertAdjacentElement('afterend', span)`.
- Créez une fonction `supprimerErreurs()` qui supprime tous les `.erreur` existants.
- Appelez `supprimerErreurs()` au début de chaque validation.

Indice : element.insertAdjacentElement('afterend', nouvelElement) insère juste après l'élément.

3.3 - Mise à jour en temps réel du range

Consignes :

- Sélectionnez l'input de type `range` (id `note`) et l'élément `<output>`.
- Écoutez l'événement `input` sur le range.

• À chaque changement, mettez à jour le contenu de `<output>` avec la valeur courante.
• Changez également la couleur de l'output selon la note : rouge si < 4, orange si < 7, vert si ≥ 7.

Exercice 4 - Interactivité de la fiche film (film_viceVersa2.html)

4.1 - Système de notation interactif

Consignes :

• Dans la section `.critiques`, ajoutez dans le HTML un bloc de notation utilisateur :

HTML à ajouter dans film.html (dans la section .critiques) :

```
<div id="notation-user">
  <h4>Donnez votre note :</h4>
  <span class="etoile" data-value="1">&#9734;</span>
  <span class="etoile" data-value="2">&#9734;</span>
  <span class="etoile" data-value="3">&#9734;</span>
  <span class="etoile" data-value="4">&#9734;</span>
  <span class="etoile" data-value="5">&#9734;</span>
  <p id="note-texte"></p>
</div>
```

CSS à ajouter dans styles.css :

```
#notation-user { margin-top: 20px; margin-bottom: 20px; }

.etoile {
  font-size: 2em;
  cursor: pointer;
  color: #f4a261;
  transition: transform 0.2s ease;
}

.etoile:hover { transform: scale(1.2); }

#note-texte { margin-top: 10px; font-weight: bold; color: #1d3557; }
```

• En JavaScript, sélectionnez toutes les étoiles (`.etoile`).

• Au survol (`mouseover`) d'une étoile, remplissez toutes les étoiles de 1 jusqu'à celle survolée (utilisez le caractère `&#9733;` pour une étoile pleine, `&#9734;` pour une vide).

• Au clic, fixez la note et affichez dans `#note-texte` : « Vous avez donné X/5 ».

• Utilisez l'attribut `data-value` pour connaître la valeur de chaque étoile : `element.dataset.value`.

Indice : L'attribut HTML data- est accessible en JS via element.dataset.nomAttribut. Par exemple, data-value='3' donne element.dataset.value === '3'.*

4.2 - Afficher/masquer les critiques

Consignes :

• Ajoutez un bouton dans le HTML, avant la liste des avis :

HTML à ajouter dans film.html (avant les <article class="avis">) :

```
<button id="toggle-critiques">Masquer les critiques</button>
```

CSS à ajouter dans styles.css :

```
#toggle-critiques {
  background: linear-gradient(135deg, #457b9d 0%, #1d3557 100%);
  color: white; border: none; padding: 10px 20px;
  border-radius: 5px; cursor: pointer; font-weight: bold;
  margin-bottom: 20px; transition: all 0.3s ease;
}
#toggle-critiques:hover { transform: translateY(-2px); box-shadow: 0 4px 8px rgba(0,0,0,0.15);
}
```

- Au clic, basculez la visibilité de tous les `.avis` (affichés/cachés).
- Changez le texte du bouton en conséquence : « Masquer les critiques » / « Afficher les critiques ».
- Utilisez une variable booléenne pour suivre l'état.

4.3 - Ajout dynamique d'un avis

Consignes :

- Ajoutez un petit formulaire à la fin de la section critiques :

HTML à ajouter dans film.html (à la fin de la section .critiques) :

```
<div id="ajout-avis">
  <h4>Laisser un avis</h4>
  <input type="text" id="avis-nom" placeholder="Votre nom">
  <textarea id="avis-texte" placeholder="Votre commentaire"></textarea>
  <button id="btn-avis">Publier</button>
</div>
```

CSS à ajouter dans styles.css :

```
#ajout-avis { background: #f8f9fa; padding: 20px; border-radius: 8px; margin-top: 20px; }
#ajout-avis h4 { margin-bottom: 15px; }
#ajout-avis input[type="text"], #ajout-avis textarea {
  width: 100%; padding: 10px; border: 2px solid #ddd; border-radius: 5px;
  font-size: 1em; margin-bottom: 10px; font-family: Arial, sans-serif;
}
#btn-avis {
  background: linear-gradient(135deg, #e63946 0%, #c53030 100%);
  color: white; border: none; padding: 10px 25px; border-radius: 5px;
  cursor: pointer; font-weight: bold; transition: all 0.3s ease;
}
#btn-avis:hover { transform: translateY(-2px); box-shadow: 0 4px 8px rgba(0,0,0,0.15); }
```

- Au clic sur « Publier », vérifiez que les deux champs sont remplis.
- Créez dynamiquement un nouvel `<article class="avis">` contenant le nom en `<h4>` et le commentaire en `<p>`.
- Ajoutez la date du jour (formatée en français) en `<em>`.
- Insérez ce nouvel avis dans la section `.critiques`, avant le formulaire d'ajout.
- Videz les champs après l'ajout.

Indice : Pour formater la date en français, utilisez :
new Date().toLocaleDateString('fr-FR', { day: 'numeric', month: 'long', year: 'numeric' })

Exercice 5 - Tri du tableau (catalogue.html)

Cet exercice vous fait manipuler les tableaux JavaScript (Array) et le DOM de manière avancée pour trier dynamiquement le tableau de films.

Modification HTML préalable

*Pour que le tri fonctionne correctement, il faut **supprimer les attributs** `rowspan` sur la colonne Genre dans le tableau de `catalogue.html`. Chaque ligne doit avoir sa propre cellule Genre. Par exemple, au lieu d'avoir un seul « Animation » fusionné sur 2 lignes, chaque ligne aura sa cellule `<td>Animation</td>`.*

Rappels utiles

```
// Trier un tableau
monTableau.sort((a, b) => a - b);     // tri numérique croissant
monTableau.sort((a, b) => b - a);     // tri numérique décroissant
monTableau.sort((a, b) => a.localeCompare(b)); // tri alphabétique

// Fonctions fléchées (arrow functions)
const double = (x) => x * 2;
const saluer = (nom) => 'Bonjour ' + nom;
```

Consignes :

- Rendez les en-têtes du tableau (`<th>`) cliquables en ajoutant `style.cursor = 'pointer'` via JavaScript.
- Au clic sur un en-tête, triez les lignes du `<tbody>` selon la colonne correspondante.
- Pour déterminer la colonne, utilisez l'index de l'en-tête cliqué (propriété `cellIndex` ou boucle).
- Triez en ordre croissant au premier clic, puis décroissant au second clic (basculement).
- Pour la colonne « Année » et « Note », faites un tri numérique. Pour les autres, un tri alphabétique.
- Ajoutez un indicateur visuel (↑ ou ↓) à côté de l'en-tête actuellement trié.

Indice : Pour déplacer les lignes triées, convertissez-les en tableau avec Array.from(tbody.querySelectorAll('tr')), triez ce tableau, puis réinsérez chaque ligne avec tbody.appendChild(row). Le DOM déplace automatiquement l'élément au lieu de le dupliquer.

Exercice 6 (Bonus) - Utilisation de fetch et de l'API TMDB

Mini-cours : Les requêtes HTTP avec fetch

JavaScript permet de communiquer avec des serveurs distants pour récupérer ou envoyer des données, sans recharger la page. C'est le principe de l'**AJAX** (Asynchronous JavaScript and XML). La fonction `fetch()` est la méthode moderne pour faire ces requêtes.

Qu'est-ce qu'une API ?

Une **API** (Application Programming Interface) est un service accessible via une URL qui renvoie des données (généralement au format **JSON**) au lieu d'une page HTML. Par exemple, l'API TMDB (The Movie Database) permet de rechercher des informations sur des films en interrogeant : `https://api.themoviedb.org/3/search/movie?query=Inception`

Le format JSON

JSON (JavaScript Object Notation) est un format texte pour représenter des données structurées. Il ressemble aux objets JavaScript. Voici un exemple de réponse de l'API TMDB :

```
{
  "id": 27205,
  "title": "Inception",
  "overview": "Dom Cobb est un voleur expérimenté...",
  "release_date": "2010-07-15",
  "vote_average": 8.369,
  "poster_path": "/ljsZTbVsrQSqZgWeep2B1QiDKuh.jpg"
}
```

Syntaxe de fetch

La fonction `fetch()` renvoie une **promesse** (Promise). Une promesse représente une opération qui prend du temps (la requête réseau). On utilise `.then()` pour définir ce qu'on fait quand la réponse arrive :

```
// Syntaxe avec .then() (promesses)
fetch('https://api.exemple.com/donnees')
  .then(function(response) {
    return response.json(); // convertit la réponse en objet JS
  })
  .then(function(data) {
    console.log(data); // utilise les données
  })
  .catch(function(erreur) {
    console.error('Erreur :', erreur);
  });
```

Explication étape par étape :

1. `fetch(url)` *envoie la requête HTTP et renvoie une promesse.*
2. *Le premier* `.then()` *reçoit la réponse brute et appelle* `.json()` *pour la convertir.*
3. *Le deuxième* `.then()` *reçoit les données JSON converties en objet JavaScript.*
4. `.catch()` *gère les erreurs (problème réseau, serveur indisponible...).*

L'API TMDB (The Movie Database)

Pour cet exercice, nous utiliserons l'**API TMDB** (The Movie Database). C'est une API gratuite et très complète, utilisée par des milliers d'applications, qui fournit des informations détaillées sur les films (titre, synopsis, note, affiche, acteurs, etc.).

Obtenir une clé API (Token) :
1. Créez un compte gratuit sur `https://www.themoviedb.org/signup`
2. Allez dans Paramètres > API (`https://www.themoviedb.org/settings/api`)
3. Récupérez votre **jeton d'accès (API Read Access Token)**

Documentation officielle :
`https://developer.themoviedb.org/reference/getting-started`

Authentification TMDB

Pour s'authentifier sur l'API, il faut ajouter un en-tête HTTP `Authorization: Bearer TOKEN` dans les options du fetch :

```
// Méthode : Token Bearer dans les en-têtes
var TOKEN = 'votre_read_access_token';

var FETCH_OPTIONS = {
  method: 'GET',
  headers: {
    'Authorization': 'Bearer ' + TOKEN,
    'Content-Type': 'application/json'
  }
};

// Utilisation :
fetch(url, FETCH_OPTIONS)
  .then(function(response) { return response.json(); })
  .then(function(data) { ... });
```

Endpoints utilisés dans cet exercice

Endpoint	URL	Description
Recherche de films	`GET /3/search/movie ?query=mot_cle&language=fr-FR`	Recherche par mot-clé. Renvoie une liste de films.
Détails d'un film	`GET /3/movie/{movie_id} ?language=fr-FR`	Détails complets d'un film (synopsis, durée, genres...).
Images	`Base URL pour les affiches : https://image.tmdb.org/t/p/w500`	Préfixe à ajouter devant poster_path pour afficher l'image.

Remarque : le paramètre `language=fr-FR` est optionnel mais permet d'obtenir les résultats en français (titres, synopsis). Sans ce paramètre, les résultats sont en anglais.

Structure de la réponse de /3/search/movie

Quand vous faites une recherche, l'API renvoie un objet avec un tableau `results` :

```
{
  "page": 1,
  "results": [
    {
      "id": 27205,
      "title": "Inception",
      "overview": "Dom Cobb est un voleur...",
      "release_date": "2010-07-15",
      "vote_average": 8.369,
      "poster_path": "/ljsZTbVsrQSqZgWeep2B1QiDKuh.jpg"
```

```
    },
    { ... }
  ],
  "total_results": 5,
  "total_pages": 1
}
```

Pour afficher l'affiche d'un film, concaténez la base URL avec `poster_path` *:*
`https://image.tmdb.org/t/p/w500` + `poster_path`
Exemple : `https://image.tmdb.org/t/p/w500/ljsZTbVsrQSqZgWeep2B1QiDKuh.jpg`

6.1 - Barre de recherche de films (index.html)

Consignes :

• Ajoutez dans le HTML une section de recherche :

HTML à ajouter dans index.html (dans le <main>, après la section films) :

```
<section class="recherche-api">
  <h2>Rechercher un film (TMDB)</h2>
  <input type="text" id="recherche-api" placeholder="Titre du film...">
  <button id="btn-recherche-api">Rechercher</button>
  <div id="resultats-api"></div>
  <div id="detail-api" style="display:none;"></div>
</section>
```

CSS à ajouter dans styles.css :

```
.recherche-api { margin-top: 40px; }

#recherche-api {
  width: 70%; padding: 10px; border: 2px solid #ddd;
  border-radius: 5px; font-size: 1em; margin-right: 10px;
}

#btn-recherche-api {
  background: linear-gradient(135deg, #e63946 0%, #c53030 100%);
  color: white; border: none; padding: 10px 25px;
  border-radius: 5px; cursor: pointer; font-weight: bold;
}
#btn-recherche-api:disabled { opacity: 0.5; cursor: not-allowed; }

#detail-api {
  background: white; padding: 20px; border-radius: 10px;
  margin-top: 20px; box-shadow: 0 4px 8px rgba(0,0,0,0.1);
}
```

• Au clic sur le bouton, récupérez la valeur du champ de recherche.

• Construisez l'URL de requête vers `https://api.themoviedb.org/3/search/movie` avec les paramètres `query` et `language=fr-FR`.

• Faites une requête `fetch(url, FETCH_OPTIONS)` vers cette URL.

• Affichez les résultats dans `#resultats-api` :

  • Pour chaque film du tableau `data.results`, créez une carte affichant le titre (`title`), l'année (extraite de `release_date`) et l'affiche (`poster_path`).

  • Si `poster_path` est `null`, affichez un texte « Pas d'affiche ».

  • Si aucun résultat (`data.results.length === 0`), affichez « Aucun film trouvé. ».

• Gérez les erreurs avec `.catch()` et affichez un message en cas de problème.

Indice : Utilisez `encodeURIComponent(recherche)` pour encoder correctement les caractères spéciaux (espaces, accents) dans l'URL.

6.2 - Fiche détaillée au clic

Consignes :

• Rendez chaque carte de résultat cliquable.

• Au clic, faites une seconde requête `fetch` vers `https://api.themoviedb.org/3/movie/{id}` en utilisant l'`id` du film cliqué.

• Affichez les détails complets dans `#detail-api` : titre original (`original_title`), synopsis (`overview`), note (`vote_average`), durée (`runtime`), genres (`genres` - c'est un tableau d'objets), date de sortie.

• Affichez les genres sous forme de liste : `data.genres.map(function(g) { return g.name; }).join(', ')`.

• Ajoutez un bouton « Fermer » pour masquer la fiche détaillée.

6.3 - Gestion du chargement et des erreurs

Consignes :

• Pendant le chargement, affichez un message « Chargement en cours... » dans `#resultats-api`.

• Désactivez le bouton de recherche pendant la requête (`button.disabled = true`) et réactivez-le une fois la réponse reçue.

• Gérez le cas où l'API renvoie une erreur en affichant un message clair à l'utilisateur.

• Ajoutez la possibilité de lancer la recherche en appuyant sur la touche Entrée (event `keydown`, vérifier `event.key === 'Enter'`).

Récapitulatif des notions couvertes

Exercice	Notions JavaScript
1 - Sélection et événements	DOM, getElementById, querySelector, querySelectorAll, addEventListener, classList, Date, conditions
2 - Manipulation du DOM	createElement, appendChild, textContent, innerHTML, style, scrollTo, window events
3 - Validation formulaire	event.preventDefault, value, trim, insertAdjacentElement, fonctions
4 - Interactivité fiche	dataset, mouseover, boucles, création dynamique, toLocaleDateString
5 - Tri du tableau	Array.from, sort, localeCompare, fonctions fléchées, cellIndex
6 - Fetch API (Bonus)	fetch, .then(), .catch(), Promises, JSON, API REST, TMDB API, requêtes HTTP, Bearer Token

Notes pédagogiques

- **Sélection du DOM** avec les méthodes standard (getElementById, querySelector)
- **Événements** avec addEventListener (click, input, submit, scroll, mouseover)
- **Manipulation du DOM** : création, suppression, modification d'éléments
- **Conditions et boucles** (if/else, for, forEach)
- **Fonctions** classiques et fonctions fléchées
- **Objets et tableaux** JavaScript (JSON)
- **fetch** et les promesses (.then/.catch) pour les appels API

Les notions avancées comme async/await, les classes ES6, ou les frameworks (React, Vue) ne sont pas abordées. Le prochain TD (TD4) portera sur PHP côté serveur pour rendre le site pleinement fonctionnel.