Mise en forme de l'interface.

Maintenant que notre première interface React est fonctionnelle, il est temps de lui donner du style. Une application sans mise en forme reste difficile à lire, peu agréable à utiliser et ne reflète pas réellement l'expérience attendue pour un projet comme CinéTech.

Dans ce TP, nous allons découvrir plusieurs manières d'ajouter du style dans une application React, du plus traditionnel au plus moderne.

Notre objectif est double :

- Comprendre les différentes méthodes possibles pour intégrer du CSS dans un projet React ;
- Utiliser TailwindCSS pour construire rapidement une interface propre et lisible.

À la fin de ce TP, notre application CinéTech aura enfin une vraie identité visuelle et une structure plus claire.

1. Les différentes façons de styliser une application

Avant de choisir une méthode, rappelons les trois grands modes d'intégration du CSS dans un projet React.

Chacune des méthodes répond à un usage précis et doit être choisie en fonction du contexte.

1.1. Le style inline

Très certainement la méthode la plus connue… et aussi la plus évitée aujourd'hui.

Le style inline consiste à appliquer directement du CSS dans le JSX.
C'est simple mais fortement déconseillé pour structurer une interface complète. Au même titre que l'écriture de styles CSS directement dans des pages HTML.

Exemple :

```
return <p style={{ color: "blue", fontSize: "20px" }}>Bienvenue</p>;
```

Explications :

- L'attribut style attend obligatoirement un objet JavaScript, et non une chaîne de caractères.
- Les propriétés CSS utilisent la notation camelCase (ex. fontSize).

Travail à réaliser :

- Ajouter un style au titre « h1 » (texte rouge et fond jaune).
- Ajouter un style aux titre « h2 » (taille de texte 20px).

La minute Git :
Envoyer votre code sur votre répertoire Github.
Nom du commit : « TP4 : style inline »

1.2. Le fichier CSS classique

C'est la méthode la plus simple et la plus proche du développement web traditionnel.
On crée un fichier .css, puis on l'importe dans le fichier JSX :

```
import './style.css';
```

Les classes s'utilisent ensuite dans du JSX :

```
return <h1 className="title">CinéTech</h1>;
```

Remarque importante :
En JSX, l'attribut class n'existe pas. Il est remplacé par className.

Travail à réaliser :

- Créer un fichier « style.css ».
- Ajouter une classe « title » coloré en vert.
- Appliquer cette classe au titre « h1 ».

⇨ *Retirer les styles inline utilisés précédemment.*

La minute Git :
Envoyer votre code sur votre répertoire Github.
Nom du commit : « TP4 : style css classique »

1.3. Le CSS Modules

Les CSS Modules permettent de limiter l'impact des classes et d'éviter les conflits de noms.
Il suffit de créer un fichier dont le nom se termine par :

```
.module.css
```

Exemple d'import :

```
import styles from './App.module.css';

return <h1 className={styles.title}>CinéTech</h1>;
```

Explications :

- Un fichier « *.module.css » est reconnu automatiquement par React
- Le nom du fichier commence classiquement par une majuscule lorsqu'il est lié à un fichier principal (ex. App.module.css).

Travail à réaliser :

- Créer un fichier « src/App.module.css ».
- Ajouter une classe « title » avec un style différent de la précédente.
- Modifier App.jsx pour utiliser cette classe.

La minute Git :
Envoyer votre code sur votre répertoire Github.
Nom du commit : « TP4 : style css modules »

1.4. Les préprocesseurs CSS

React permet aussi d'utiliser des préprocesseurs CSS tels que Sass, Less ou Stylus etc. Ils offrent la possibilité de générer dynamiquement des fichiers CSS tout en simplifiant l'écriture de celui-ci. Les préprocesseurs vont par exemple ajouter des variables, des conditions ou encore des boucles pour en optimiser l'écriture. La finalité de ces préprocesseurs est d'organiser efficacement le code source d'une application, mais de ne générer qu'un seul fichier CSS compressé.

Nous n'utiliserons pas de préprocesseur dans CinéTech, mais cela peut être une piste d'exploration intéressante pour celles et ceux qui souhaitent approfondir.

1.5. Le choix pour CinéTech

Nous venons de découvrir trois manières d'intégrer du style dans une application React. Chacune a son utilité, mais pour un projet moderne comme CinéTech, nous avons besoin d'un système flexible, rapide à mettre en place, et modulable.

C'est pourquoi nous allons intégrer TailwindCSS, une bibliothèque très populaire qui utilise des classes utilitaires pour produire du style immédiatement.

Dans la prochaine partie, nous allons installer TailwindCSS et vérifier que son intégration fonctionne correctement.

2. Installation de TailwindCSS dans CinéTech

TailwindCSS propose des classes prêtes à l'emploi pour structurer rapidement une interface.

Il permet :

- De styliser une interface sans écrire du CSS complexe,
- D'obtenir un résultat propre en quelques minutes,
- De rester concentré sur la logique React.

Commençons par installer le paquet grâce à NPM.

Dans un terminal du nuage (à la racine de cinetech-react), exécuter :

```
> npm install -D tailwindcss@3 postcss autoprefixer
> npx tailwindcss init -p
```

Explications :

- La première commande installe TailwindCSS et les dépendances nécessaires
- La deuxième génère deux fichiers de configuration : « `tailwind.config.js` » et « `postcss.config.js` ».

Nous allons ouvrir le fichier « tailwind.config.js » et modifier le tableau content :

```
/** @type {import('tailwindcss').Config} */
export default {
  content: [
    "./index.html",
    "./src/**/*.{js,ts,jsx,tsx}",
  ],
  theme: {
    extend: {},
  },
  plugins: [],
}
```

Nous pouvons maintenant créer un nouveau fichier « src/index.css » puis y ajouter :

```
@tailwind base;
@tailwind components;
@tailwind utilities;
```

Explications :
Ces lignes permettent d'importer les différents fichiers CSS dans notre application.

Puis importons ce fichier dans notre main.jsx :

```
import './index.css';
```

Avant de tester, nous devons redémarrer le serveur :

```
> npm run dev
```

La police d'écriture de la page devrait changer une fois que le serveur a bien été redémarré.

Travail à réaliser :

- Modifier temporairement le <h1> avec la classe « text-red-500 ».
- Vérifier que le texte passe bien en rouge.

Tailwind est maintenant intégré à notre projet.
Nous allons pouvoir styliser CinéTech avec une approche simple et fluide.

3. Mise en forme du titre et de la structure principale

Nous allons structurer notre page en appliquant des styles génériques.
Le style proposé ici reste volontairement simple. Il est tout à fait possible de le personnaliser à votre guise.

Documentation Tailwind :
https://tailwindcss.com/docs/installation/using-vite

Avant de continuer :
Retirer tous les styles inline et les fichiers CSS utilisés précédemment, afin de partir sur une base propre

Appliquons un style à notre titre principal dans « App.jsx » :

```
// …
<h1 className="text-4xl font-bold text-center my-6">
  CinéTech
</h1>

// …
```

Explications :

- text-4xl : taille de police ;
- font-bold : texte en gras ;
- text-center : centrage horizontal ;
- my-6 : marges verticales.

Pour la zone principale, nous allons simplement centrer le contenu et laisser des gouttières sur le côté.

Toujours dans le fichier « App.jsx » nous devons ajouter les classes CSS suivantes à « <main> » :

```
// …
<main className="max-w-7xl mx-auto mt-6">
// …
```

Explications :

- max-w-7xl : largeur maximale ;
- mx-auto : centrage automatique à gauche et à droite ;
- mt-6 : marge au-dessus.

Pour le titre de notre section :

```
// …
<h2 className="text-2xl font-bold mb-4">Films à l'affiche</h2>
// …
```

La structure est maintenant posée.
Passons à la partie la plus visuelle du TP : afficher des cartes de films avec un rendu moderne.

4. Mise en forme de la liste des films

Il nous reste maintenant à styliser notre liste des films et à rendre l'affichage de chacun d'eux moins brut.

Prenons le code suivant :

```
<ul>
        {moviesList.map((movie) => (
          <li key={movie.id}>
            <h3>{movie.title}</h3>
            <p>
              {movie.year} — {movie.director}
            </p>
            <p>Note : {movie.rating}</p>
          </li>
        ))}
</ul>
```

Nous allons commencer par appliquer un effet « carte de film » sur chaque élément liste :

```
// …
<li key={movie.id} className="bg-gray-100 rounded-xl p-4 shadow">
// …
```

Maintenant nous allons faire en sorte que les cartes de film s'affichent en grille :

```
// …
<ul className="grid grid-cols-1 sm:grid-cols-5 gap-6">
// …
```

Explications :
Ces classes CSS permettent de créer une grille responsive.

Travail à réaliser :

- Pour chaque « li », afficher l'image du film.
- Appliquer les classes CSS suivantes :

```
w-full h-64 object-cover rounded
```

La minute Git :
Envoyer votre code sur votre répertoire Github.
Nom du commit : « TP4 : ajout film image »

Travail à réaliser :

- Mettre le titre du film en taille « XL » et avec une épaisseur de police semi-grasse.
- Mettre l'élément paragraphe contenant l'année et le réalisateur en taille « SM » avec une couleur de texte grise en nuance 600.
- Mettre une marge de 2 sur le haut des éléments nécessaires.

Notre interface CinéTech prend enfin forme avec une présentation claire et visuelle.
Nous avons manipulé Tailwind à petite dose, mais suffisamment pour mettre en forme une vraie interface moderne.

5. Challenge final du TP

Pour conclure ce TP, nous allons mettre en pratique tout ce que nous avons appris, dans un exercice un peu plus complexe.

Nous souhaitons mettre un film en avant au-dessus de la liste complète.

Travail à réaliser :

- Ajouter une nouvelle section « Film du moment ».
- Afficher dans cette section
  - Le titre du film en grand
  - L'affiche du film en grande taille
  - La note en couleur
  - Le petit résumé
- Styliser la section pour la rendre visuellement attractive.

La minute Git :
Envoyer votre code sur votre répertoire Github.
Nom du commit : « TP4 : section film du moment »

Trop facile ?!

Travail à réaliser :

- Ajouter et styliser une barre de navigation.
- Ajouter et styliser un pied de page.
- Personnaliser le site à votre goût.

La minute Git :
Envoyer votre code sur votre répertoire Github.
Nom du commit : « TP4 : personnalisation du style »

6. Conclusion du TP

Nous venons de donner vie à CinéTech en y ajoutant une véritable mise en forme visuelle.

Nous avons appris que React laisse une grande liberté concernant l'intégration du CSS, et que TailwindCSS permet d'obtenir rapidement un rendu professionnel tout en restant focalisé sur la logique de l'application.

Nous avons posé les bases d'une interface propre, structurée et agréable à parcourir.

Dans le prochain TP, nous irons plus loin en introduisant une nouvelle notion essentielle : la séparation de l'interface grâce aux composants React.