Développement d’applications web client

TD n°7 – Axios & introduction à vue-cli

On inclut Axios via un CDN dans l’en-tête, vue préconise d’utiliser cette librairie avec son framework.
Axios permet d’exécuter des requêtes XHR à l’aide de promises ce qui permet d’utiliser directement les méthodes .then .catch & .finally

La documentation d’Axios est disponible ici :
https://axios-http.com/

Une structure particulière du code HTML utilisant des directives v-if de vue.js est très pratique dans le cas de requêtes XHR.
Voir l’exemple : *axios-methode-workflow.html*

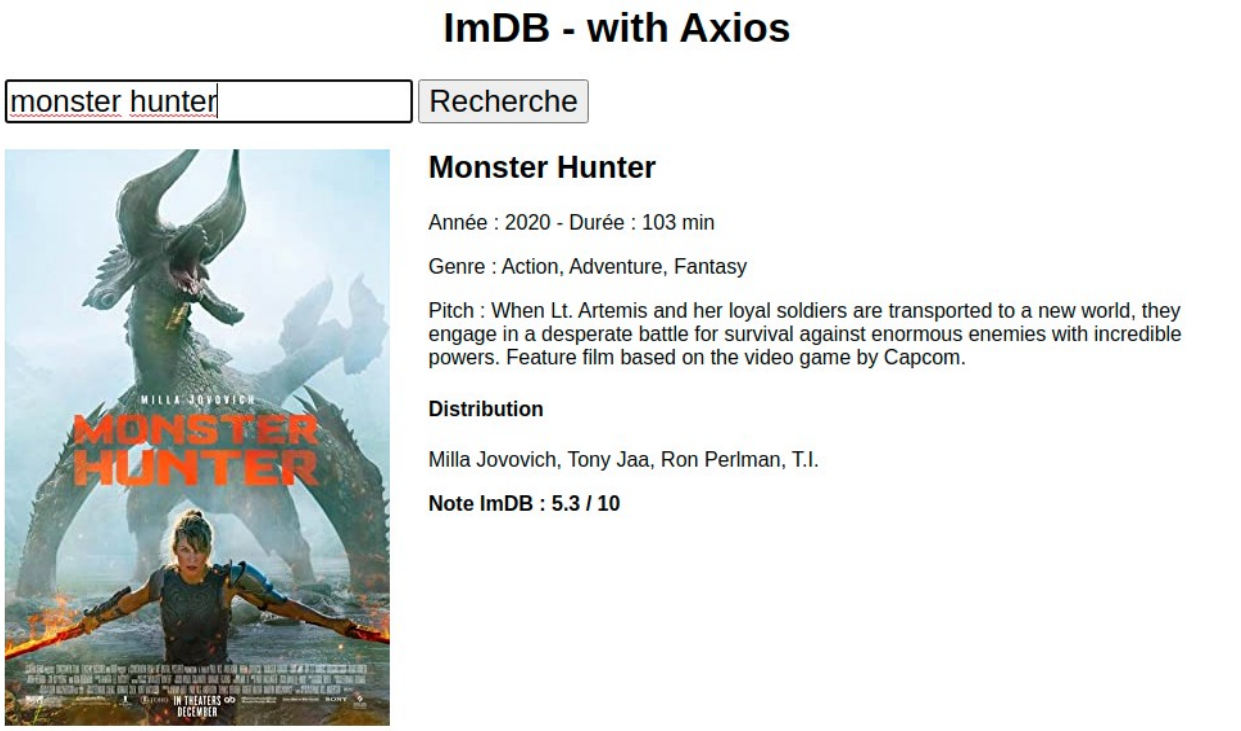
Note : Lorsque vous devez réaliser une interface, prenez le temps d’analyser 2 ou 3 interfaces similaires qui se trouvent sur des sites connus et à forte fréquentation. Cela vous permet d’identifier les éléments importants qui doivent être affichés, l’ergonomie et les conventions de mise en page.
Exemple : le bouton ou input de connexion se trouve toujours en haut à droite.

Exercice Recherche dans un dictionnaire
Réalisez une interface de recherche de lexèmes
On pourra choisir la langue (fr ou en)
Afficher les différents sens, les définitions et la nature du lexème
Utilisez un framework CSS pour l’affichage
API pour la recherche de mots : https://dictionaryapi.dev/

Note : Affichez toujours l’URL générée (dans un console.log) que vous envoyez à l’API afin de la verifier directement dans votre browser. Vous pouvez afficher la réponse JSON dans un viewer en ligne, JS Viewer :
https://jsonformatter.org/json-viewer

Exercice iMDB
Réalisez une interface de recherche de film à partir du titre sur ImDB, ainsi que l'affichage des 8 infos les plus importantes du film (dont l'affiche).
Dans l’interface on gardera la partie recherche toujours accessible.
Vous pouvez rajouter un framework CSS en option.
La création d'une clé d'API est obligatoire (et rapide)
http://www.omdbapi.com/

Vous trouverez dans l’exemple de réalisation de cet exercice quelques bonnes pratiques concernant la réalisation d’interfaces utilisateur abouties :
- Mettre le focus dans le premier input au chargement de la page
- Gérer la validation de la saisie avec la touche [Entrée]
- Gérer tous les cas d’erreur et de retour de l’API et afficher des messages clairs et correspondants à l’utilisateur



Note : j’ai mis un exemple pour une fois ;)

Introduction à vue-cli

Vue-cli est orienté développement d'applications. L'ensemble du projet est compilé donc on oublie complètement le SEO (Search Engine Optimization)
Ne pas l'utiliser pour un site web qui doit être référencé sur les moteurs de recherche.

Pour travailler avec vue-cli on a besoin de node.js version 8.9 minimal > 10 conseillé.
Installation de node :

```
sudo apt install npm
```

Installation de vue-cli :

```
sudo npm install -g @vue/cli
```

Vérifier l'install :

```
vue --version
```

Install de cli-init :

```
sudo npm install -g @vue/cli-init
```

Créer un projet :

```
vue init webpack-simple premiere-app
```

Un dossier du même nom est créé, ensuite pour terminer la préparation du projet :

```
cd premiere-app
npm install
```

Une structure de projet est définie, c'est le dossier `src` qui nous intéresse :
- le point d'entrée dans l'application est le fichier App.vue
- On crée un dossier components dans lequel tous les fichiers components (fichiers .vue) seront enregistrés
- le dossier Assets contiendra toutes les images

La structure de base d’un fichier .vue

on y trouve les sections suivantes

```
<template> … </template>

<script>
export default { }
…
</script>

<style> … </style>
```