

TP Vue.js

I. Présentation

1. Objectifs:

L'objectif de ce TP est de vous familiariser avec les concepts clés de Vue.js tout en créant une application web interactive sur le thème des superhéros. Vous apprendrez à récupérer des données à partir d'une API externe, à les afficher dynamiquement, et à créer des fonctionnalités interactives.

2. Contraintes:

- Thème : Les superhéros
- Utilisation de l'API : Nous utiliserons l'API Superhero qui est accessible à l'adresse suivante : <https://github.com/rtomczak/superhero-api/tree/0.3.0/api>
- Pour récupérer la liste de tous les superhéros : ajoutez **/all.json** à l'adresse de base.
- Pour obtenir les informations d'un superhéros spécifique (par exemple, l'id 1) :
 - Superhéros : **/id/1.json**
 - Superpouvoirs : **/powerstats/1.json**
 - Apparence : **/appearance/1.json**
 - Biographie : **/biography/1.json**
 - Images : Pour obtenir différentes tailles d'images, vous pouvez utiliser les chemins suivants :
 - Toute petite image : **/images/xs/1-a-bomb.jpg**
 - Petite image : **/images/sm/1-a-bomb.jpg**
 - Moyenne image : **/images/md/1-a-bomb.jpg**
 - Grande image : **/images/lg/1-a-bomb.jpg**

II. Présentation de la page d'accueil - Affichage de tous les superhéros

1. Présentation

Dans cette page d'accueil de notre application sur les superhéros, nous allons présenter une liste complète de tous les superhéros disponibles. Nous utiliserons l'API Superhero pour récupérer les données pertinentes et les afficher de manière dynamique à l'aide du framework Vue.js.

2. Qu'est-ce qu'un fichier json ?

Un fichier JSON (JavaScript Object Notation) est un format de données léger et couramment utilisé pour l'échange de données entre applications. Il s'agit d'un format texte basé sur une syntaxe d'objet JavaScript qui permet de représenter des structures de données simples ou complexes.

Un fichier JSON est généralement composé de paires clé-valeur, où chaque clé est une chaîne de caractères entourée de guillemets doubles, et la valeur peut être de différents types de données, tels que des nombres, des chaînes de caractères, des booléens, des tableaux (listes ordonnées) ou des objets (collections de paires clé-valeur).

Voici un exemple simple d'un fichier JSON :

```
{
  "nom": "John Doe",
  "âge": 30,
  "ville": "Paris",
  "intérêts": ["musique", "sport", "lecture"],
  "employeur": {
    "nom": "ABC Company",
    "poste": "Développeur"
  }
}
```

Dans cet exemple, nous avons un objet JSON avec différentes paires clé-valeur. Par exemple, "nom" est une clé avec la valeur "John Doe", "âge" est une clé avec la valeur 30, "intérêts" est une clé avec un tableau de valeurs, et "employeur" est une clé avec un objet imbriqué contenant les clés "nom" et "poste".

Les fichiers JSON sont largement utilisés dans le développement d'applications web et mobiles pour échanger des données avec des serveurs, stocker des configurations ou des paramètres, et pour de nombreux autres cas d'utilisation. Ils offrent une structure de données simple, lisible par les humains et facile à manipuler à l'aide de langages de programmation tels que JavaScript.

3. Affichage du tableau

Tous les personnages seront mis dans un tableau **superheros**

Étape N°1 : Créez une instance d'application Vue.js et déclarez une variable **superheros** comme une liste vide dans les données de l'application. Cette liste sera utilisée pour stocker les informations des superhéros. Ensuite affichez cette liste (qui est actuellement vide).

Le résultat doit ressembler à cela :

Liste des superhéros

[]

4. Récupération des données

a) mounted

La méthode **mounted()** est utilisée pour exécuter du code après que l'application Vue a été montée sur l'élément cible dans le DOM, ce qui permet notamment de faire des opérations supplémentaires, comme des requêtes HTTP pour obtenir des données externes, et de mettre à jour les données de l'application en conséquence.

Étape N°2 : Ajoutez la ligne suivante à la fin du head

```
<script src="https://unpkg.com/axios@latest"></script>
```

Étape N°3 : Ajoutez la méthode suivante :

```
mounted() {
  axios.get('https://cdn.jsdelivr.net/gh/rtomczak/superhero-api@0.3.0/api/all.json')
    .then(response => {
      this.superheros = response.data; // Récupérer les données des superhéros
    })
    .catch(error => {
      console.log(error);
    });
}
```

Vous devez obtenir cet affichage :

Liste des superhéros

```
[{"id": 1, "name": "A-Bomb", "slug": "1-a-bomb", "powerstats": {"intelligence": 38, "strength": 100, "speed": 17, "durability": 80, "power": 24, "combat": 64}, "appearance": {"gender": "Male", "race": "Human", "height": [ "6'8", "203 cm" ], "weight": [ "980 lb", "441 kg" ], "eyeColor": "Yellow", "hairColor": "No Hair" }, "biography": { "fullName": "Richard Milhouse Jones", "alterEgos": "No alter egos found.", "aliases": [ "Rick Jones" ], "placeOfBirth": "Scarsdale, Arizona", "firstAppearance": "Hulk Vol 2 #2 (April, 2008) (as A-Bomb)", "publisher": "Marvel Comics", "alignment": "good" }, "work": { "occupation": "Musician, adventurer, author; formerly talk show host", "base": "-"}, "connections": { "groupAffiliation": "Hulk Family; Excelsior (sponsor), Avengers (honorary member); formerly partner of the Hulk, Captain America and Captain Marvel; Teen Brigade; ally of Rom", "relatives": "Mario Chandler-Jones (wife); Polly (aunt); Mrs. Chandler (mother-in-law); Keith"
```

b) Explications

L'instruction **axios.get('https://cdn.jsdelivr.net/gh/rtomczak/superhero-api@0.3.0/api/all.json')** est une requête HTTP GET utilisant la bibliothèque Axios pour récupérer des données à partir de l'URL spécifiée. Plus spécifiquement, cette requête GET est envoyée à l'URL **'https://cdn.jsdelivr.net/gh/rtomczak/superhero-api@0.3.0/api/all.json'**.

Cette URL représente un point d'accès à un fichier JSON contenant des informations sur des superhéros.

L'utilisation de **axios.get()** indique que nous souhaitons effectuer une requête GET pour obtenir les données à partir de cette URL.

La bibliothèque Axios facilite l'envoi de requêtes HTTP et la gestion des réponses.

Une fois que la requête est effectuée, il est possible de traiter la réponse en utilisant les méthodes **then()** et **catch()** pour gérer les cas de succès et d'erreur respectivement. Dans le code que vous avez fourni, la méthode **then()** est utilisée pour récupérer les données de réponse, tandis que la méthode **catch()** est utilisée pour afficher une éventuelle erreur dans la console.

5. Affichage de la liste

Étape N°4 : Affichez les id ainsi que les noms de tous les personnages comme ceci :

Liste des superhéros

- 1 - Nom : A-Bomb
- 2 - Nom : Abe Sapien
- 3 - Nom : Abin Sur
- 4 - Nom : Abomination
- 5 - Nom : Abraxas
- 6 - Nom : Absorbing Man

Étape N°5 : Remplacez la liste par des paragraphes :

Liste des superhéros

- 1 - Nom : A-Bomb
- 2 - Nom : Abe Sapien
- 3 - Nom : Abin Sur
- 4 - Nom : Abomination
- 5 - Nom : Abraxas

6. Avec du CSS

Étape N°6 : Utilisez bootstrap pour embellir votre page :

Liste des superhéros



A-Bomb	1
Abe Sapien	2
Abin Sur	3
Abomination	4
Abraxas	5

Utilisation des classes bootstrap :

- **my-4** pour ajouter une marge en haut et en bas du titre principal.
- **list-group** pour créer une liste de superhéros stylisée.
- **list-group-item** pour chaque élément de la liste.
- **list-group-item-action** pour rendre les éléments cliquables.
- **d-flex** pour aligner horizontalement le nom du superhéros et son ID.
- **w-100** pour que le contenu s'étende sur toute la largeur.
- **justify-content-between** pour espacer le nom du superhéros et son ID
- le nom est un **h5** et l'id est **small**.

Étape N°7 : Ajoutez les images entre le nom et l'id.

Liste des superhéros

A-Bomb		1
Abe Sapien		2


- Conseils : la propriété **personnage.images.md** pour récupérer l'URL de l'image correspondante et la classe **rounded-circle** pour donner à l'image une forme arrondie. Et pour le texte **align-items-center** pour les centrer verticalement. **width** de 50 pixels

7. *Détail*

Étape N°8 : Ajoutez une case à cocher : Afficher les pouvoirs




☒ Afficher les pouvoirs

Étape N°9 : Si cette case est sélectionnée, alors les pouvoirs sont affichées ainsi

A-Bomb		Intelligence: 38 Force: 100 Vitesse: 17 Durabilité: 80 Pouvoir: 24 Combat: 64	1
--------	---	---	---

8. Recherche par nom

Étape N°10 : Ajoutez une recherche par nom :

	<h3>Batgirl</h3>	63
	<h3>Batgirl IV</h3>	66
	<h3>Batgirl VI</h3>	68

Aide

- Il faut utiliser la directive **v-model**
- La valeur saisie dans le champ de recherche est liée à la propriété `searchQuery` dans l'objet `data` de l'application `Vue.js`.
- Ajout une propriété calculée **filteredSuperheros** qui retourne les superhéros filtrés en fonction de la valeur de recherche saisie.
- Utiliser la méthode `.filter` sur `this.superheros`

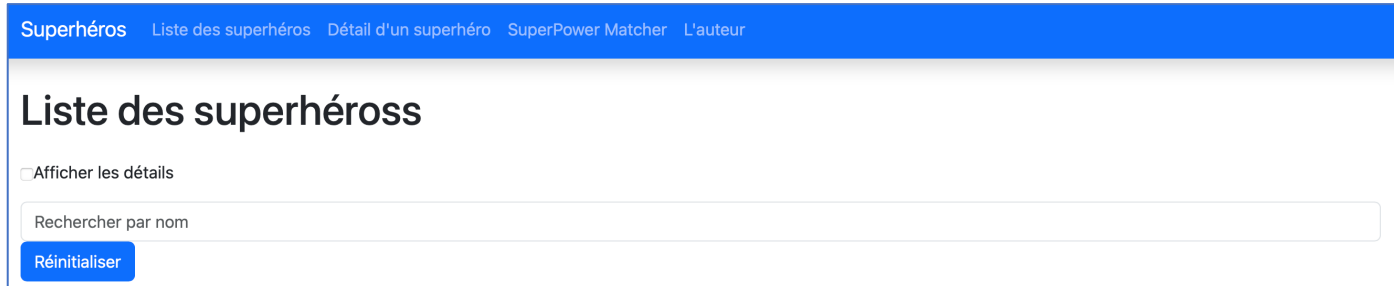
Étape N°11 : Ajoutez un bouton qui réinitialisera la recherche

Réinitialiser

III. Barre de navigation

1. Création de la barre

Étape N°12 : Mettez en place une barre de navigation, avec les liens et les pages correspondantes



2. Même barre pour toutes les pages

Étape N°13 : Créez un fichier HTML séparé appelé "navbar.html" (ou tout autre nom de votre choix) et copiez-y le code de votre barre de navigation actuelle

Étape N°14 : sur chaque page utilisez une balise `<div id= "____">` pour inclure le contenu du fichier "navbar.html".

Étape N°15 : charger le contenu du fichier "navbar.html" dans la balise `<div>` avec l'id "navbar-placeholder". Vous pouvez utiliser la méthode `fetch` pour charger le contenu du fichier et l'insérer dans la balise `<div>`.

```
fetch('navbar.html')
  .then(response => response.text())
  .then(data => {
    document.getElementById(____).____ = data;
  })
```

IV. Détail d'un personnage

Étape N°16 : Dans la page index.html, lorsqu'un utilisateur clique sur un super-héros, l'application doit naviguer vers une page de détail pour ce super-héros (détail.html) en fournissant l'ID du personnage.

Étape N°17 : Puis cette page affichera tous les détails du superheros

V. SuperPowerMatcher

Étape N°18 : **SuperPower Matcher** : Sur cette page, ajoutez un questionnaire où les utilisateurs peuvent sélectionner les superpouvoirs qu'ils aimeraient avoir. Lorsqu'ils soumettent le questionnaire, l'application doit afficher une liste de super-héros qui ont ces superpouvoirs.

[Superhéros](#)
[Liste des superhéros](#)
[Détail d'un superhéro](#)
[SuperPower Matcher](#)
[L'auteur](#)

SuperPower Matcher

Intelligence 90

Force 80

Vitesse 94

Rechercher

Résultats :

- Beyonder
- General Zod
- Living Tribunal
- Man of Miracles
- Mister Mxyzptlk
- One-Above-All
- Power Girl
- Superboy-Prime
- Supergirl
- Superman