Rapport d’introduction au framework Vue JS

Installation d’un projet

L’un des prérequis pour démarrer un projet Vue JS est d’être à l’aise avec l’utilisation d’un Bundler et d’avoir node installé aussi sa machine. La commande pour lancer le projet est :

```
npm create vue@latest
```

Après avoir lancé cette commande , il nous ait demandé d’installer des packages de bases assez intéressant dans notre projet. A ce niveau, nous avons la possibilité d’accepter ou de décliner. Après initialisation du projet, on exécute les commandes suivantes :

```
cd nomProjet
nom install
npm run dev
```

Cycle de vie d’un composant

Le cycle de vie d'un composant Vue.js est une série d’étapes (création, mise à jour, destruction).

Ces hooks permettent d'exécuter du code personnalisé à des moments spécifiques, comme onMounted après l'insertion du composant dans le DOM, ou Unmounted juste avant sa suppression. Ils sont essentiels pour gérer l'état, interagir avec le DOM et effectuer des opérations de nettoyage.

Principales phases et hooks

Création

- **beforeCreate** : Appelé juste après l'initialisation de l'instance, avant que les données réactives et les propriétés ne soient traitées.
- **Created** : Appelé après que l'instance ait terminé de traiter ses options d'état, mais avant qu'elle ne soit montée dans le DOM.
- **BeforeMoun**t : Appelé juste avant que le composant ne soit monté dans le DOM.
- **Mounted** : Appelé une fois que le composant a été rendu et inséré dans le DOM. Idéal pour les interactions avec le DOM ou les appels API.

Mise à jour

- **beforeUpdate** : Appelé juste avant que le DOM ne soit mis à jour en réponse à un changement d'état réactif.

- **Updated** : Appelé après la mise à jour du DOM.

Destruction

- **beforeUnmount** : Appelé juste avant que le composant ne soit démonté.
- **Unmounted** : Appelé après que le composant ait été démonté et effacé du DOM. Parfait pour le nettoyage des écouteurs d'événements ou des abonnements.

les hooks : explication, usage et utilité

Les hooks dans Vue.js permettent aux développeurs d'exécuter du code personnalisé (par exemple, interagir avec le DOM ou gérer des requêtes asynchrones) à des étapes clés, offrant ainsi plus de contrôle et de flexibilité.

Explication et utilité

Hooks du cycle de vie : Ce sont des fonctions qui sont exécutées automatiquement à différentes phases du cycle de vie d'un composant, de sa création à sa destruction.

Utilité : Ils permettent d'intervenir à des moments précis, comme l'initialisation de variables, l'ajout de code à l'élément DOM après sa création (onMounted), la mise à jour après un changement de données (onUpdated) ou le nettoyage des ressources lors de la destruction du composant (onUnmounted).

Hooks de directives personnalisées : Ils sont utilisés pour ajouter un comportement à des directives personnalisées.

Utilité : Ils permettent d'accéder à des informations comme l'élément DOM auquel la directive est liée (el), sa valeur (binding.value), l'ancien et le nouveau nom (binding.oldValue), ou encore les arguments et modificateurs passés à la directive.

watch vs computed: explication, utilité, usage et différence

Computed est utilisé pour calculer des valeurs dérivées de manière réactive et mise en cache, tandis que **watch** est utilisé pour déclencher des actions complexes ou asynchrones en réponse à un changement de donnée. La principale différence réside dans l'objectif : computed sert à créer une nouvelle valeur, et **watch** sert à exécuter du code (comme une requête HTTP).

Propriétés computed

Utilité : Calculer des valeurs qui dépendent d'autres propriétés réactives. Elles sont idéales pour les états dérivés.

Usage : Créer une nouvelle propriété dans le template à partir de données réactives. Par exemple, calculer un prix total à partir de price et quantity.

Observateurs watch

Utilité : Exécuter du code en réponse à un changement dans une propriété spécifique, surtout pour des opérations complexes ou asynchrones.

Usage : Réagir à un changement de donnée.Le watch ne retourne pas de valeur mais exécute une fonction de rappel (callback) qui peut effectuer des actions.

Axios, localStorage et vuex: explication, utilité

Axios, localStorage et Vuex sont trois outils clés dans les applications web, chacun ayant un rôle distinct : **Axios** pour les requêtes HTTP vers un serveur, **localStorage** pour un stockage de données côté client persistant, et **Vuex** pour la gestion de l'état global d'une application Vue.

Axios

Utilité : C'est un client HTTP qui facilite les requêtes réseau (par exemple, pour récupérer ou envoyer des données à une API) depuis le navigateur. Il prend en charge les promesses (Promises), permet d'intercepter les requêtes et les réponses, de transformer les données automatiquement (comme le JSON) et de protéger contre les attaques XSRF.
On l'utilise pour la communication entre l'application frontend et le serveur backend.

LocalStorage

- **Utilité** : Il s'agit d'un mécanisme de stockage local dans le navigateur pour sauvegarder des données sous forme de paires clé-valeur. Les données stockées dans localStorage persistent même après la fermeture du navigateur et sont accessibles via des méthodes comme setItem() , getItem() et clear() (pour effacer). Il est utile pour stocker les préférences utilisateur, mettre en cache des données d'API ou sauvegarder l'état d'une application pour une persistance entre les sessions.

Vuex

Utilité : C'est un gestionnaire d'état centralisé pour les applications Vue.js. Il permet de maintenir un "état" unique et prévisible pour l'ensemble de l'application, de manière centralisée et plus structurée qu'avec localStorage pour la gestion des données réactives. Il est idéal pour les applications complexes où les données doivent être partagées et modifiées par de nombreux composants. Il est recommandé pour les nouveaux projets, bien que Vuex soit toujours maintenu, Pinia est le nouveau gestionnaire d'état recommandé pour les nouvelles applications Vue