Services.

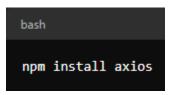
Step-by-Step Guide

1. Create a Vue.js Project

First, create a new Vue.js project using the Vue CLI or npm init vue@latest. For this example, let's assume the project is already created.

2. Install Axios

Install Axios for making HTTP requests:



3. Create a Service File

Create a directory named services inside the src folder, and then create a new file named recipeService.js inside the services directory.

4. Define the Service

In recipeService.js, define a function to fetch data from the "chinese-food-db" API.

```
// src/services/recipeService.js
import axios from 'axios'; 55.6k (gzipped: 20.6k)
// Base URL and API Key for the Chinese Food DB API
const API_URL = 'https://chinese-food-db.p.rapidapi.com';
const API KEY = 'YOUR RAPIDAPI KEY'; // Replace with your RapidAPI key
// Axios request options
const options = {
 method: 'GET',
 url: API URL,
 headers: {
    'X-RapidAPI-Host': 'chinese-food-db.p.rapidapi.com',
    'X-RapidAPI-Key': API KEY
};
// Function to fetch recipes from the API
export const getRecipes = async () => {
 try {
   const response = await axios.request(options);
   return response.data; // Return the data from the API response
 } catch (error) {
   console.error('Error fetching recipes:', error);
   throw error; // Throw the error to handle it in the component
```

5. Create a Component to Use the Service

Create a component to fetch and display recipes using the service.

```
<script>
import { ref, onMounted } from 'vue'; 558.2k (gzipped: 177.7k)
import { getRecipes } from '../services/recipeService';
export default {
 name: 'RecipeList',
 setup() {
   // Define a reactive variable to store the recipes
   const recipes = ref([]);
   // Function to fetch recipes and update the reactive variable
   const fetchRecipes = async () => {
     try {
       recipes.value = await getRecipes();
     } catch (error) {
       console.error('Error fetching recipes:', error);
   onMounted(fetchRecipes);
     recipes // Return the reactive variable to use in the template
</script>
```

```
<style>
/* Add some basic styling */
ul {
    list-style: none;
    padding: 0;
}

li {
    margin-bottom: 20px;
}

img {
    display: block;
    margin-bottom: 10px;
}
</style>
```

6. Import and Use the Component in App.vue

Import and use the RecipeList component in your main App.vue.

Explanation

1. Service Definition:

- recipeService.js contains the logic for fetching data from the API using Axios.
- getRecipes is an asynchronous function that makes an API request and returns the data.

2. Component Setup:

- RecipeList.vue is a Vue component that displays a list of recipes.
- We import getRecipes from recipeService.js.
- We use ref to create a reactive variable recipes to store the fetched data.
- fetchRecipes is an asynchronous function that calls getRecipes and updates recipes.
- onMounted is a lifecycle hook that calls fetchRecipes when the component is mounted.

3. Template:

 The template displays a list of recipes with their images, titles, and difficulty levels.

By organizing the code into services and components, we keep the logic separate and make it easier to manage and understand.

Key JavaScript Concepts

1. export

The export keyword is used to make a variable, function, or class available for import in other files. This helps in modularizing the code.

In our service file, we use export to make the getRecipes function available to other files:

```
javascript

export const getRecipes = async () => {
    // Function implementation
};
```

2. async and await

async and await are used to handle asynchronous operations (like API calls) in a more readable way than traditional promises.

- async: This keyword is used before a function to make it asynchronous. It allows the use of await inside the function.
- await: This keyword is used to wait for a promise to resolve. It can only be used inside an async function.

In our example:

```
javascript

export const getRecipes = async () => {
   try {
     const response = await axios.request(options);
     return response.data;
   } catch (error) {
     console.error('Error fetching recipes:', error);
     throw error;
   }
};
```

async makes getRecipes an asynchronous function.

await waits for the Axios request to complete and returns the result.

3. ref([])

ref is a function from the Vue Composition API used to create a reactive reference to a value. It makes a value reactive, meaning Vue will track its changes and update the DOM accordingly.

In our example:

```
javascript
import { ref } from 'vue';
const recipes = ref([]);
```

ref([]) creates a reactive reference to an empty array.

• recipes will be updated reactively when new data is fetched.

4. onMounted

onMounted is a lifecycle hook in the Vue Composition API. It runs a function when the component is mounted (added to the DOM).

In our example:

```
javascript
import { onMounted } from 'vue';
onMounted(fetchRecipes);
```

onMounted(fetchRecipes) runs the fetchRecipes function when the RecipeList component is added to the DOM.

Component Setup:

ref([]): Creates a reactive array recipes.

- onMounted(fetchRecipes): Calls fetchRecipes when the component is mounted.
- fetchRecipes: An asynchronous function that calls getRecipes and updates recipes.

Summary

- export: Makes a function available to other files.
- async: Defines an asynchronous function.
- await: Waits for a promise to resolve.
- ref([]): Creates a reactive reference to an array.
- onMounted: Runs a function when the component is mounted.

This modular approach separates the data-fetching logic (service) from the UI logic (component), making the code cleaner and easier to maintain.