什么是Axios？为什么要使用Axios?

是一个基于Promise的网络请求库，使用它最大的好处就是它支持请求拦截，和响应拦截。这样我们可以用它，在正式发起网络请求前，做一些验证，例如本地到底有没有存储token认证字符串，如果没有，就没必要向我们的后端发起请求了，这样可以在一定程度上减轻后端服务器的压力。另外我们还可以用响应拦截做一下对服务器return回来的数据做统一、简单的处理。以提升开发效率。总之，Axios是一个非常好用的网络请求库！

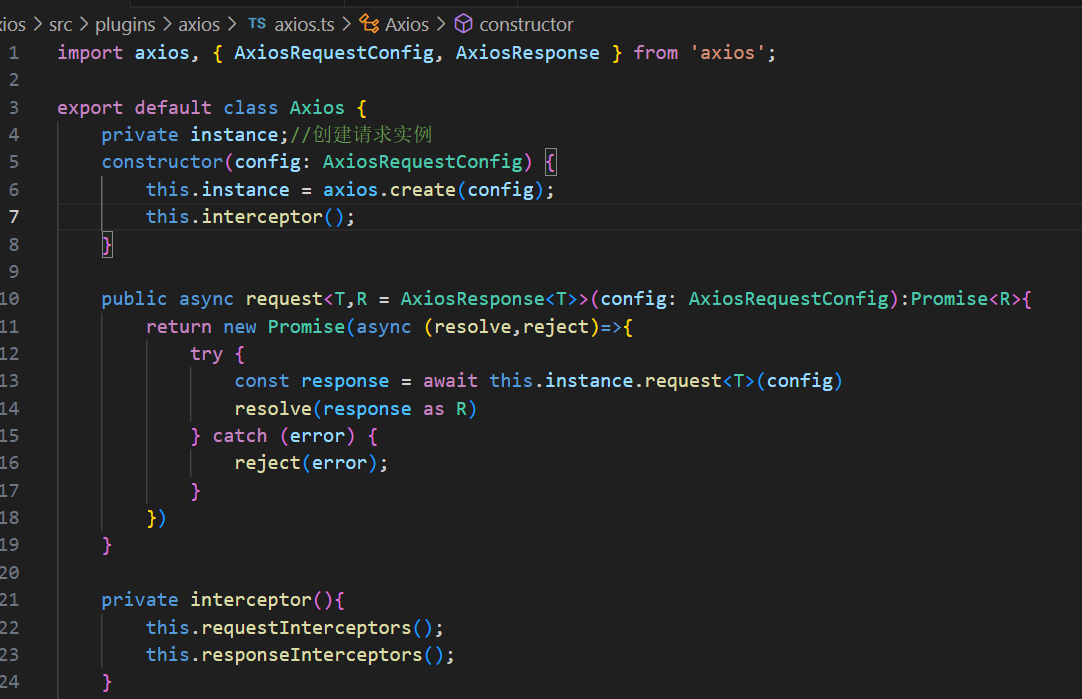
前言：

这个小demo主要是为了展示如何封装axios网络请求组件，demo中的请求地址，和数据都是使用mock.js自动生成的，关于如何在vite中使用mockJs，可以直接访问mockJs的github主页，非常简单。同样我们Axios也需要提前安装。

如何封装Axios网络请求组件？

整体思路是使用class封装出一个Axios类，在这个类中定义一个public方法，外部通过使用这个public方法，发送网络请求，在Axios类中我们可以将请求和响应拦截都定义为私有方法，在这些方法中对请求和响应做统一管理。最后我们向外暴露这个类即可。

我们来看一下具体的代码：



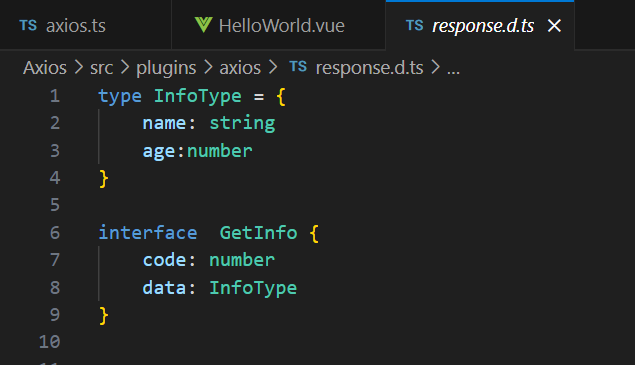
使用ts对Axios响应数据做类型声明----提升开发效率

封装Axios组件其实不是一件难事，难事是我们希望使用ts对响应的数据做自动的类型声明，这样好提升开发效率，因为通过网络请求得到的数据，默认是一个联合类型any，这样的话我们在使用网络请求返回的数据的时候，是没有代码提示的。这其实是一个“一劳永逸”的方法而已。

我们先来看看，后端的接口我们是怎么定义的，顺便瞅一眼返回的数据长啥样。



观察了接口响应的数据后，其实呢，我们就可以在vite给我们提供的\*.d.ts类型声明文件中，定义对应的类型。



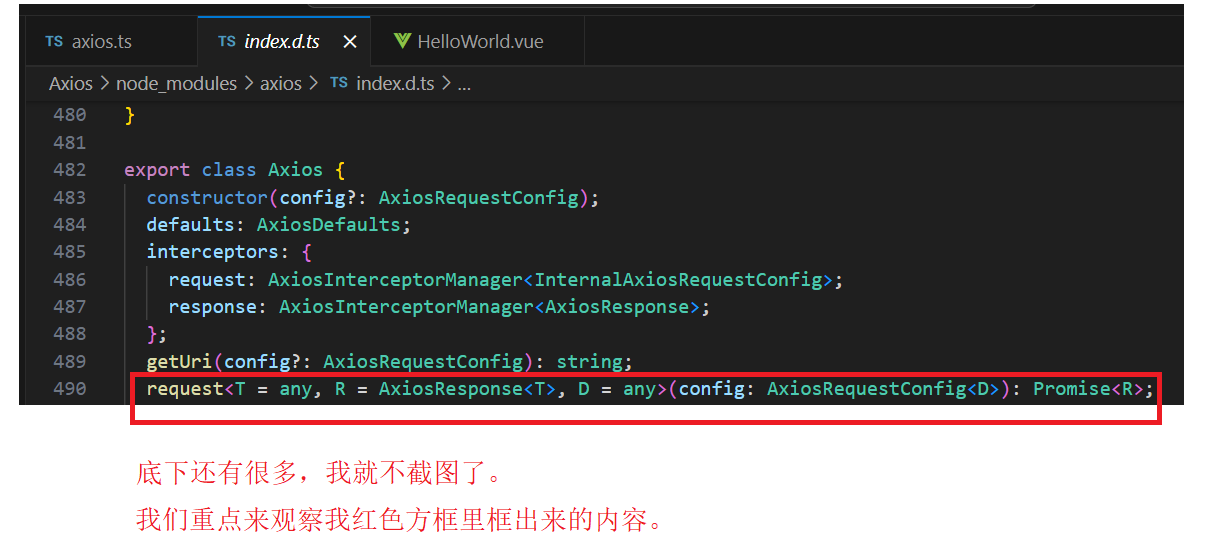
Get请求返回数据的类型声明。

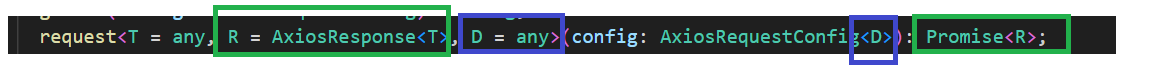
有关什么是vite的类型声明文件，可以参看vite官网，非常简单的东西昂，它就是一个类型声明文件，在ts中可以直接使用类型声明文件中的类型。

使用ts泛型完成对响应数据的类型声明。

这里需要你对ts的泛型有一定的认识，没有也没关系，这里我会详细说明的。以期大家都能理解。

我们直接点进Axios的类型声明文件中，我们先来看看它是怎么定义request()方法的。



我们分两部分来解释：  


第一部分（绿色框框圈出来的地方）：

Promise<R>是对Promise中resolve()中的数据的限制。

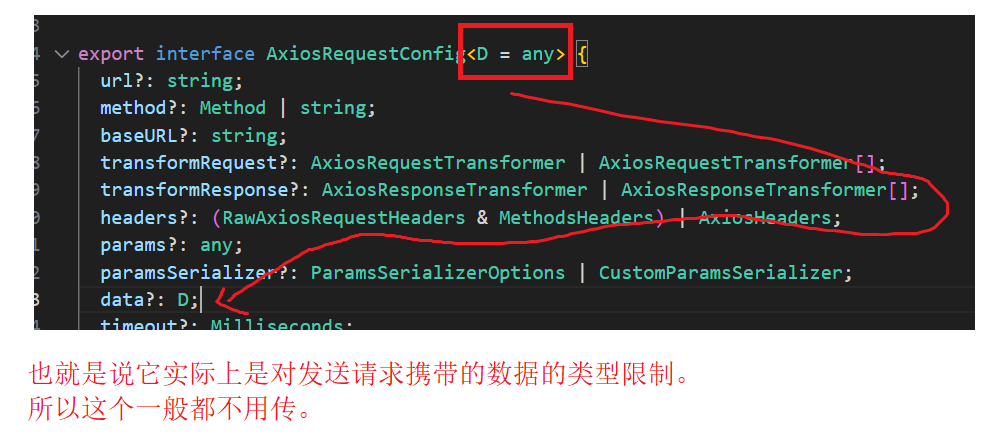
在使用request的时候，除非你一开始传入了R这类型，否则的话它默认等于AxiosResponse<T>泛型函数的结果（泛型函数返回的也是一个类型，它是ts给我们提供的一种用于类型运算的特殊方法。）我们来看一下所谓的AxiosResponse<>泛型函数是啥。



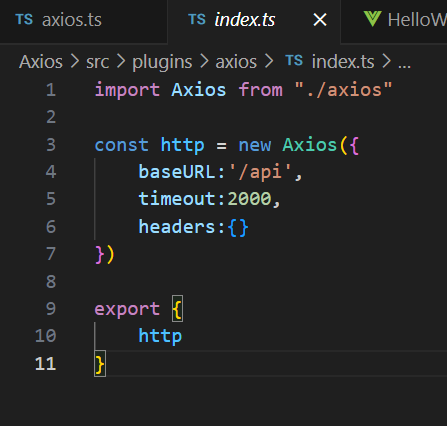
也就是说我可以在使用request的时候传递一个与返回数据结果对应的类型，这样就完成了对返回数据的类型声明。所以根据response返回的结果我们可以这样定义类型声明。

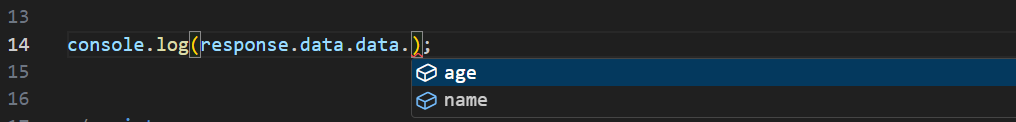


第二部分（蓝色框框圈出来的地方）：  
D如果不传，默认就是any。它对request中的config参数进行类型限制。我们直接来看在类型声明文件中它是怎么定义AxiosRequestConfig泛型的。



写到这，其实我们就已经大致完成了Axios模块的封装，和对Axios请求得到的数据的类型声明。最后我们来看一眼，如何使用我们封装好的Axios模块，以及类型声明是否可用。





看到没有，类型提示已经来了。

页面效果：也是相当的完美！

