# axios 拦截 , 页面跳转, token 验证(经常用于判断用户是否登录)

第一步: [路由](https://router.vuejs.org/zh-cn/api/component-injections.html" \t "https://blog.csdn.net/u012878818/article/details/_blank) 多添加一个自定义字段 requireAuth

path: '/repository',

name: 'repository',

meta: {

requireAuth: true, // 添加该字段，表示进入这个路由是需要登录的

},

component: Repository

第二步:

router.beforeEach((to, from, next) => {

if (to.meta.requireAuth) { // 判断该路由是否需要登录权限

if (store.state.token) { // 通过vuex state获取当前的token是否存在

next();

}

else {

next({

path: '/login',

query: {redirect: to.fullPath} // 将跳转的路由path作为参数，登录成功后跳转到该路由

})

}

}

else {

next();

}

登录拦截到这里就结束了吗？并没有。

这种方式只是简单的前端路由控制，并不能真正阻止用户访问需要登录权限的路由。(可手动在浏览器地址栏输入没有权限的路由)

还有一种情况便是：当前token失效了，但是token依然保存在本地。

这时候你去访问需要登录权限的路由时，实际上应该让用户重新登录。

这时候就需要结合 http 拦截器 + 后端接口返回的http 状态码来判断。

第三步:  拦截器 (要想统一处理所有http请求和响应，就得用上 axios 的拦截器。)

每次跳页面, 都要获取新路由对应的html页面, 这时候可以用axios的http拦截

每次路由跳转, 都先让后台验证一下token是否有效, 在http头添加token,

当后端接口返回 401 Unauthorized（未授权） ，让用户重新登录。

[关于Autorization](http://www.cnblogs.com/findumars/p/5107574.html" \t "https://blog.csdn.net/u012878818/article/details/_blank)   使用之后会忽略cookie的token,  削弱了安全性, 可以配合https

*// http request 拦截器*

axios.interceptors.request.use(

config => {

if (store.state.token) { *// 判断是否存在token，如果存在的话，则每个http header都加上token*

config.headers.Authorization = `token ${store.state.token}`;

}

return config;

},

err => {

return Promise.reject(err);

});

*// http response 拦截器*

axios.interceptors.response.use(

response => {

return response;

},

error => {

if (error.response) {

switch (error.response.status) {

case 401: 401 旌旗 灵医 , 只用[授权] 旌旗的医生 才是 灵医

*// 返回 401 清除token信息并跳转到登录页面*

store.commit(types.LOGOUT);

router.replace({

path: 'login',

query: {redirect: router.currentRoute.fullPath}

})

}

}

return Promise.reject(error.response.data) *// 返回接口返回的错误信息*

});

完整的方法见 /src/http.js .

通过上面这几步，就可以在前端实现登录拦截了。

登出 功能也就很简单，只需要把当前token清除，再跳转到首页即可。