Unknown custom element: <router-view> - did you register the

这种错误是在路由中没有使用vue.use(Router)

<input v-model.number="msg" @input="output"/>{{typeof msg}}

    <!-- //这是number类型 -->

output(e){

      console.log(typeof e.target.value)//在这里是string类型

    },

<https://blog.csdn.net/qq_42238554/article/details/86592295>事件修饰符

动态路由

<https://blog.csdn.net/weixin_41646716/article/details/80450873>插槽

组件中 data 为什么是函数

因为组件是用来复用的，JS 里对象是引用关系，这样作用域没有隔离

怎样理解单向数据流

这个概念出现在组件通信。父组件是通过 prop 把数据传递到子组件的，但是这个 prop 只能由父组件修改，子组件不能修改，否则会报错。子组件想修改时，只能通过 $emit 派发一个自定义事件，父组件接收到后，由父组件修改。

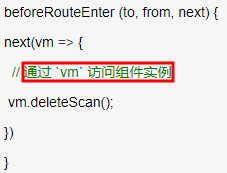
1、全局的钩子

beforeEach（to，from，next）

beforeResolve   
afterEach（to，from，next）

2、组建内的导航钩子

beforeRouteEnter、beforeRouteUpdate、beforeRouteLeave。



default只有在case匹配失败的时候才会执行(react-route路由)



大大的坑，记得用require

cnpm install --save vue-i18n

keepAlive就是在路由跳转之后不会再调用目标页面的生命周期

keepalive控制部分组件就行

<keep-alive>

      <router-view  v-if="$route.meta.keepAlive"/>

    </keep-alive>

    <router-view  v-if="!$route.meta.keepAlive"/>

query会出现在地址栏上

异步加载组件，store刷新之后被还原

情况一：

import ElementUI from 'element-ui'  
import 'element-ui/lib/theme-default/index.css'  
Vue.use(ElementUI)

情况二：

import 'element-ui/lib/theme-chalk/index.css'

拦截：

import Vue from 'vue'

import router from './router/index.js'

Vue.use(router)

axios.interceptors.response.use((response)=> {

    console.log(response.data.code==10001)

    if(response.data.code==10001){

        console.log('登录失效了')

        router.replace('/login')

    }

    return response;

}, function(error) {

    return Promise.reject(error);

});

go(-1): 原页面表单中的内容会丢失；

history.go(-1):后退+刷新；

history.go(1) ：前进

back(): 原页表表单中的内容会保留；

history.back():后退 ；

history.back(0) 刷新；

history.back(1)：前进

this.$bus.$off("toDoor")

    this.$bus.$on("toDoor",()=>{

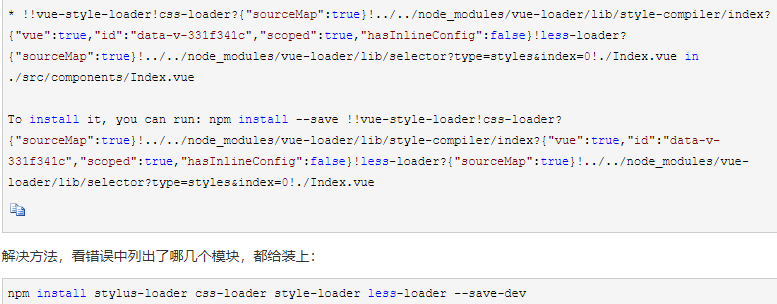
      //关掉先

    })

 router.replace('/login')//在http.js中使用router

store.state.navContent//在main.js中使用store

config.headers.common['XX-Token'] = sessionStorage.getItem('token');



然后提示需要loader,cnpm i vux-loader -D

//将module.exports = {

改为var originalConfig = {}

