前端项目难点亮点

1.如何防止重复提交

一般使用的是防抖和节流，节流函数通过控制每次时间执行的时间间隔，控制短时间多次执行方法。防抖函数是推迟每次事件执行的时间减少不必要的查询。但是网络慢的时候，还是会重复提交，没有显示状态，用户不知道有没有真的提交。所以就给按钮添加一个加载状态，查了发现el-button自带了loading属性，传参的时候传一个submit函数，是一个Promise,promise状态改变的时候把loading状态改成false。然后点击按钮会有加载动画，加载的时候，按钮是禁用的。

2.控制ajax执行先后顺序

一个按钮发送2个ajax请求，不会按顺序，因为是异步请求，浏览器可以并行执行，执行快慢看的是响应数据量大小和后台逻辑的复杂程度，为了保证顺序，就是是最后的结果，需要改成同步，ajax请求后台数据，获取数据之后渲染到页面，就需要同步。请求加一个async:false,就可以让ajax会同步执行。但是请求时间比较长需要loading层显示等待状态，但是浏览器渲染线程和js线程互斥，执行js的时候页面渲染会阻塞掉，让ajax函数后面的代码还有渲染线程停止，就算dom操作语句是发起请求的前一句，也会被阻塞，出现假死卡顿现象，所以可以引入JQuery中的对象deferred进行封装异步函数，对多个deferred对象进行并行化操作，当所有deferred对象都得到解决就执行后面添加的回调。

3.解析数据

填写信息的页面，返回的时候填写信息需要留存，要用vue的keep-alive实现

4.axios用post请求数据会被拦截，传不到后端

ajax请求可以拿到数据，axios就拿不到，因为axios的post默认参数格式是字符串，传给后端的数据需要用请求拦截器做处理。可以引入qs库，对data进行处理：qs.stringify。Qs装axios的时候自动安装了。

5.路由懒加载

需要的时候进行加载，把不同路由对应的组件分成不同代码块，路由被访问才加载对应组件。路由会定义很多页面，页面打包后放到单独的js文件会导致非常大，懒加载把页面进行划分，需要时才加载，减少首页加载速度，懒加载主要就是把对应组件打包成js代码块进入首屏不用加载过度的资源，从而减少首屏加载速度。就是用import，在路由配置的router.js，import设置好的组件，from后面写的是组件的路径

6.实现从详情页返回列表页保存上次加载的数据和自动还原上次的浏览位置。

vue2中提供了keep-alive。keep-alive是Vue提供的一个抽象组件，用来对组件进行缓存，从而节省性能，由于是一个抽象组件，所以在v页面渲染完毕后不会被渲染成一个DOM元素，当组件在keep-alive内被切换时组件的**activated、deactivated**这两个生命周期钩子函数会被执行

被包裹在keep-alive中的组件的状态将会被保留，例如我们将某个列表类组件内容滑动到第100条位置，那么我们在切换到一个组件后再次切换回到该组件，该组件的位置状态依旧会保持在第100条列表处。如果要每次进入组件时页面初始位置都是顶部，可以用路由提供的基础功能scrollBehavior

```
const router=new VueRouter({
        routes:[
            {
                path:"/",
                component:Home
            }
        ],
        scrollBehavior(to,form,savedPosition){
        //scrollBehavior方法接收to，form路由对象
        //第三个参数savedPosition当且仅当在浏览器前进后退按钮触发时才可用
        //该方法会返回滚动位置的对象信息，如果返回false，或者是一个空的对象，那么不会发生滚动
        //我们可以在该方法中设置返回值来指定页面的滚动位置，例如:
         return {x:0,y:0}
        //表示在用户切换路由时让是所有页面都返回到顶部位置
        //如果返回savedPosition,那么在点击后退按钮时就会表现的像原生浏览器一样，返回的页面会滚动过到之前按钮点击跳转的位置，大概写法如下:
         if(savedPosition){
            return savedPosition
         }else{
           return {x:0,y:0}
         }
         //如果想要模拟滚动到锚点的行为:
         if(to.hash){
           return {
             selector:to.hash
           }
         }
    }
})
```

7.Storage封装

Storage存储在浏览器端不参与和服务器的通信，本身有api,localStorage跨页面传参，sessionStorage保存临时数据，防止刷新页面后丢失参数。本身api只是简单的key/value形式，且只存储字符串，需要人工转为json,也不能单个清空，所以对Storage封装，便于更好的操控Storage.向Storage添加参数和模块

4）Proxy实现跨域（vue项目)

在vue.config.js里面设置好proxy,里面写'/api'{}作为拦截，里面写好target:后端的接口网址，changeOrigin:true，pathRewrite:{}

9.断网处理

断网时会更新vue中network的状态，根据network状态判断是否需要加载断网组件，断网情况就加载断网组件，不加载对应页面组件，点击刷新，就跳转refresh页面然后立刻返回来实现重新获取数据，所以新建refresh.vue,在beforeRouteEnter钩子种返回当前页面。

```
<template>
    <div id="app">
        <div v-if="!network">
            <h3>我没网了</h3>
            <div @click="onRefresh">刷新</div>
        </div>
        <router-view/>
    </div>
</template>

<script>
    import { mapState } from 'vuex';
    export default {
        name: 'App',
        computed: {
            ...mapState(['network'])
        },
        methods: {
            // 通过跳转一个空页面再返回的方式来实现刷新当前页面数据的目的
            onRefresh () {
                this.$router.replace('/refresh')
            }
        }
    }
</script>
```

```
// refresh.vue
beforeRouteEnter (to, from, next) {
    next(vm => {
        vm.$router.replace(from.fullPath)
    })
}
```

10.scoped时修改子组件样式

vue文件正常样式写在</script>中，会被自动加上一个[data-v-xxxx]属性，但是第三方组件内部标签没有编译为这个属性，所以不能修改这个第三方组件。 为了父组件不影响子组件，用scoped,有一个方法是给第三方组件写class,然后在公共css或者当前页面写一个没有scoped的style,直接在里面修改第三方组件的样式，但是存在全局污染和命名冲突，约定的那个特殊的命名方式可以避免命名冲突。查找资料发现可以用深度选择器解决。深度选择器用/deep/，用到ElemntUI,又有预处理器就可以用.::v-deep。vue中过多使用scoped导致页面打包文件体积增大。通常能写在index中的样式尽量写在index中，我们可以通过在index样式中通过外层组件添加唯一class来区分组件+ `第三方样式` 来实现了类似于scoped的效果，又方便修改各种第三方组件的样式。

11.跳转页面后停止定时器

通过$once事件侦听器在定义完定时器之后的位置清除定时器

```
const timer = setInterval(() =>{
    // 某些定时器操作
}, 500);
// 通过$once来监听定时器，在beforeDestroy钩子可以被清除。
this.$once('hook:beforeDestroy', () => {
    clearInterval(timer);
})
```

vue的3个监听事件，□□(□□□□□□□□□:□□□□□□ | □□□□□,□□□□□□□□)监听当前实例的自定义事件，可以由*on*(*eventName*:*string* | *Array*,*callback*)监听当前实例的自定义事件，可以由emit触发，回调函数接收所有传入时间触发函数的额外参数，$once(eventName: string, callback)只监听一个自定义事件，只触发一次，触发后监听器移除

12.v-model进行父子组件双向数据绑定

v-model是语法糖，可以用来实现全局弹窗组件，v-model控制弹窗的显示隐藏，可以在子组件用model选项来绑定值：

```
// model选项用来避免冲突
// prop属性用来指定props属性中的哪个值用来接收父组件v-model传递的值
// 例如这里用props中的show来接收父组件传递的v-model值
// event：为了方便理解，可以简单理解为父组件@input的别名，从而避免冲突
// event的值对应了你emit时要提交的事件名，你可以叫aa，也可以叫bb，但是要命名要有意义哦！！！
model: {
    prop: 'show',
    event: 'changed'
},
props: {
    // 由于model选项中的prop属性指定了，所以show接收的是父组件v-model传递的值

    show: {
        type: Boolean,
        default: false
    }
},
methods: {
    confirm () {
        // 双向数据绑定父组件传递的值
        // 第一个参数，对应model选项的event的值，你可以叫aa，bbb，ccc，起名随你
        this.$emit('changed', false)
    }
}
```

12.页面加载的时候发现页面空白

1）JS异常

在头部加载JS，导致页面渲染被阻塞了

2）客户端请求异常

无效请求，错误路径。vue路由配置错误，比如没有配置路由，路由指向页面是空白页，配置了2个重复路由，app.vue没有防止路由占位符。

3）服务器异常

无法正常找到服务器资源，或者服务器死机（502）

4）网络问题

DNS解析异常（没办法解析出IP地址），连接超时，请求资源较大，设置连接时长，网速慢得时候无法下载完页面资源，导致页面空白。CDN服务器异常，导致上面的资源无法正常进行/加载。

13.解决vuex持久化

情景时列表页跳转到详情页，详情页是新窗口，2个窗口都用到vuex state,比如共享同一个id数组，修改state数据之后，详情页不能实时更新state数据，只能用缓存解决，比如localStorage，也有组件vuex-persistedstate，把vuex数据动态更新成storage。

14.使用history模式后访问内容页，刷新会404

需要后端重定向配置服务器。

15.菜单权限用动态添加路由addRoutes解决

有一个公共路由，登录后获取权限，得到需要动态添加的路由表，把路由添加到router里。实现方式是提前定义好完整的路由表，然后跟后台传输的权限做对比，过滤出一个路由权限表，再用addRoutes动态添加到路由里。然后根据过滤出的路由权限表渲染侧边栏。

16.vue项目中用v-for 循环本地图片， 图片不显示

需要把图片放到static文件夹，或者用requires动态引入文件。