VUEUI Unit06

处理刷新首页时滚动位置的更新问题

基于 `Vue` 的滚动行为解决刷新滚动位置的修改。

基于 `keepAlive` 实现首页的保活

在 `router/index.js` 合适的路由中，添加路由meta信息，指定哪些路由组件需要保活：

```
{
    path: 'index',
    component: () => import('../views/Index.vue'),
    meta: {
        keepAlive: true
    }
},
```

在 `App.vue` 中，使用 `keepAlive` 组件包裹 `router-view` 。被包裹的路由组件将会保活。

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

如上配置即可实现首页的保活，但是，当跳转到详情页后，在滚动时会触发 `mintui` 的无限滚动指令监听方法。解决方法如下：

当首页开启了 `keepAlive` ，将会解锁两个生命周期方法， `activated` `deactivated` 。

```
activated() {
    this.isLoading = false;
},
deactivated() {
    this.isLoading = true;
},
```

实现注册业务

业务流程：

在注册页面，填写表单；

点击快速注册，验证表单；

验证成功后，发送注册请求，提交用户信息，执行注册业务；

1. 注册成功，跳转到登录；
2. 注册失败，弹窗提示。

实现登录业务

业务流程：

在登录页面，填写表单；

点击登录，验证表单；

验证成功后，发送登录请求，提交用户信息，执行登录业务；

1. 登录成功，跳转到首页；
2. 登录失败，弹窗提示。

基于 Vuex 登录成功后在首页提示用户信息

Vuex

`state` 用于定义存在 `vuex` 中的状态信息

`mutations` 用于定义修改 `state` 所需要的方法

`actions` 用于定义异步方法，执行完异步任务后，可以调用 `mutations` 修改 `state` 。

`Vuex` 的核心功能就是在合适的时间点，存数据、取数据。

实现步骤：

1. 在 `store/index.js` 中，向state对象中存入一些信息，用来全局保存用户登录状态。

   ```
   state:{
       isLogin: true,
       name: 'zs'
   }
   ```

   在页面中引用：

   ```
   <div slot="right" v-if="$store.state.isLogin">
       欢迎：{{$store.state.name}}
   </div>
   ```

2. 在vuex的 `mutations` 声明一个方法，用于登录成功后修改 `state` ：

```
mutations: {
    /** 定义一个方法 当登录成功后修改state
     *  state: vuex将会自动传入state对象，方便操作变量
     *  newname: 该参数是调用者携带的自定义参数
     */
    loginOK(state, newname){
        state.isLogin = true
        state.name = newname
    }
},
```

如何调用`mutations`中声明的方法：

```
// commit()方法用于请求vuex，执行mutations中定义的loginOK
this.$store.commit('loginOK', 'zhangsan')
```

`Actions`

在`actions`中定义函数，**异步**完成任务得到结果后，将结果更新到`state`。

```
actions:{
    login(store, userObj){
        // actions中执行异步任务
        执行登录().then(res=>{
            // 异步任务得到结果后，不能直接操作state，应该调用store.commit方法
            // 让mutations直接修改state。
            store.commit('loginOK', userObj.name)
        })
    }
}
```

`vuex`可以在不刷新页面的前提下保存多页面的共享数据。但是一旦刷新，意味着页面将重新加载，整个`Vue`容器也将会重新加载，`Vuex`中存储的数据也将不复存在，全部初始化。也就无法保存登录状态信息。

如果希望持久化保存用户数据（刷新不销毁，甚至客户端关机重启依然存在），推荐使用`HTML5`新特性中的`WebStorage`来解决。

`WebStorage`

`WebStorage`是`HTML5`提供的可以让前端持久化缓存数据的存储空间。包含两种对象：

1. `sessionStorage`：这一块存储空间存储的数据单浏览器会话生效。重启浏览器即销毁。
2. `localStorage`：这一块存储空间存储的数据永久有效。

`WebStorage`相关`API`

存数据：

```
sessionStorage.setItem('name', '张三')
localStorage.setItem('age', '15')

let user = {"id":1,"name":"zs","age":15}
sessionStorage.setItem('user', JSON.stringify(user))
```

读数据：

```
sessionStorage.getItem('name')  -->  '张三'
localStorage.getItem('age')   -->  '15'

sessionStorage.getItem('user')  -->  JSON字符串
JSON.parse(sessionStorage.getItem('user'))  -->  JS对象
```

删除数据：

```
sessionStorage.removeItem('name')
localStorage.removeItem('name')
```

清空数据：

```
sessionStorage.clear()
localStorage.clear()
```

音频与视频

`HTML5` 提供了相关标签支持在网页中实现音频与视频的播放。

音频标签

音频标签支持的文件格式有：`WAV`、`MP3`、`ogg`。

音频标签的简单使用方法：

```
<audio src="../assets/xxxx.mp3" controls></audio>
```

音频标签的标准使用方法：

```
<audio controls>
    <source src="xxx.mp3" type="audio/mpeg"/>
    <source src="xxx.wav" type="audio/wav"/>
    <source src="xxx.ogg" type="audio/ogg"/>
    什么破浏览器，换一个吧~
</audio>
```

`audio`的常用属性

```
<audio  controls     是否显示播放器控制面板
        src=""
        autoplay     是否在加载完毕后自动播放
        muted        是否静音
        loop         是否单曲循环
        preload="预加载模式"></audio>
```

`preload`：

1. `none` 不预加载
2. `metadata` 只预加载元数据
3. `auto` 尽可能的加载音频数据

视频标签

简写方式：

```
<video src=""          视频文件的链接路径
       controls
       autoplay
       muted
       loop
       width=""        视频播放器的宽度
       height=""       视频播放器的高度
       poster=""       海报帧图片路径
></video>
```

音视频相关的 DOM 操作

我们可以通过`Javascript`获取音视频的`DOM`对象。

可以通过访问对象的属性，调用对象的方法来访问音视频播放状态，控制音视频的播放。

```
<audio id="audio" src="./let.mp3"></audio>
<script>
    let audio = document.getElementById('audio')
    audio.duration   属性  返回时长
    audio.play()     方法  播放视频
</script>
```

查看 `DOM` 相关文档:

`https://www.w3school.com.cn/tags/html_ref_audio_video_dom.asp`