28 聊天列表页面 UI 开发

更新时间：2019-09-23 10:23:51

> 生活的理想，就是为了理想的生活。
>
> ——张闻天

聊天列表页是用来存储我们自己个其他人的聊天列表数据的页面，我们在这个页面可以进行简单的文字匹配搜索，来查询以往含有关键字的聊天数据。下面就进入开发吧。

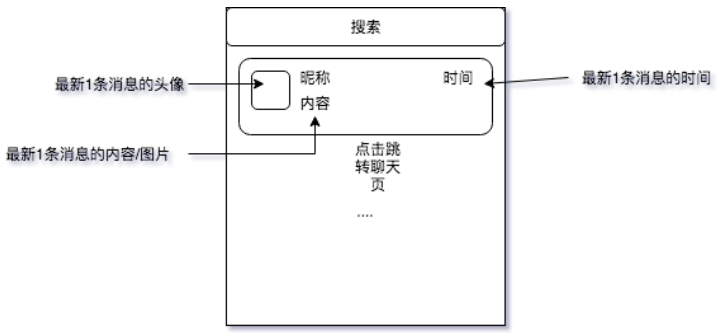
本章节完整源代码地址，大家可以事先浏览一下：

Github

页面逻辑梳理

整个聊天列表页面主要由顶部的搜索框和下面的聊天信息列表来组成，点击搜索可以根据搜索内容，筛选查看当前聊天记录，整个页面是可以滚动的，有点类似于微信的聊天记录页面。

页面逻辑图：



页面UI图：

整个页面的重点在于：

1. 搜索框和下面聊天数据的联动。
2. 聊天列表元素的 UI 布局和展示。

下面，就进入具体逻辑的开发吧。

开发顶部搜索框

在前端项目的 views 文件夹下新建 chatlist 文件夹，同时新建 index.vue，作为聊天列表页的主页面。

然后，我们新建顶部的搜索框相关的UI代码，代码如下：

```
<template>
  <div class="container">
    <navHeader title="我的私信"/>
    <div class="weui-search-bar" id="searchBar" :class="searchBarClass">
      <form class="weui-search-bar__form">
        <div class="weui-search-bar__box">
          <i class="weui-icon-search"></i>
          <input v-model="keyword" type="search" class="weui-search-bar__input" placeholder="搜索" required @input="searchChat($event)">
          <a href="javascript:" @click="clearSearch" class="weui-icon-clear"></a>
        </div>
        <label class="weui-search-bar__label">
          <i class="weui-icon-search"></i>
          <span>搜索</span>
        </label>
      </form>
      <a href="javascript:" class="weui-search-bar__cancel-btn">取消</a>
    </div>

  </div>
</template>
```

上面代码中，我们主要完成了：

1. 顶部的搜索UI，使用的是 weui 的 form 的 searchBar 组件相关的组件和样式。
2. 我们给 input 添加了 v-model 来动态获取数据。

下面，别忘了在 mounted 方法中对 searchBar 组件进行初始化，代码如下：

```
mounted () {
  weui.searchBar('#searchBar')
},
```

另外，我们需要增加的逻辑就是给 <input> 绑定 oninput 事件，来实现我们的搜索功能，在之前的章节已经讲过 oninput 事件 onchange 事件相关的知识点，这里就不在赘述，我们给 <input> 绑定v-model=keyword，同时将keyword存入vuex的 stroe，目的是为了在页面返回时，可以记录到当前搜索的值，否则就会丢失。

将 keyword 放在 computed 中，代码如下：

```
computed: {
  keyword :{
    get:function(){
      return this.$store.state.keyword;
    },
    set:function(newValue){
      // this.$store.state.keyword = newValue;
      this.$store.dispatch('setKeyword', newValue)
    }
  }
},
```

这里我们使用vuex的黄金搭配 computed 计算属性来获取 keyword ，但是需要注意的是，我们再给 computed 计算属性设置 v-model 时，需要提供 set 方法，因为对于 <input> 的 v-model ，一旦值改动就需要实时的变化 keyword ，如果没有 set 方法，就会报错。

在vuex里修改store的状态的方法

看到注释里的那段 this.$store.state.keyword = newValue; 代码了么，这里普及一个知识点：

在vuex里修改store的状态，采用action和直接修改state有何区别？

1. 理论上，如果需要修改 store 里的一个 state 值，例如上面的 keyword，直接采用 this.$store.state.keyword = newValue 和 this.$store.dispatch('setKeyword', newValue) 都是可以达到效果的，而且能够生效。
2. 直接修改 state 不需要新增 mutations， action，直接操作即可，而后者需要添加 mutations， action 。
3. 大多数情况下，vuex 的官方是不推荐直接修改 state 的值，这样不利于代码的维护，而且 vue 调试工具无法跟踪到。另外这里我们只是简单修改 state 的一个值，逻辑相对简单，但是如果后续逻辑复杂，修改 state 前需要很多逻辑，而且这些逻辑在很多地方都会用到，这样我们采用 mutations， action 就更加高效维护性高一些了。
4. 如果我们在创建 vuex 时启用了严格模式 strict:true，那么我们代理里就不能直接修改 state 的值了，只能采用 mutations， action 的方式：

```
export default new Vuex.Store({
  strict: process.env.NODE_ENV !== 'production',//严格模式，只能用action来修改state
...
})
```

但是严格模式不建议在生存环境中使用，会消耗更多的性能。

列表UI开发

我们接着在前端项目的 views 文件夹下的 chatlist 文件夹下的 index.vue 中，添加聊天列表相关的UI逻辑，代码如下：

```
<div class="content-list">
  <div class="weui-loadmore" v-show="loading">
    <i class="weui-loading"></i>
    <span class="weui-loadmore__tips">正在加载</span>
  </div>
  <div class="weui-loadmore weui-loadmore_line weui-loadmore_dot" v-show="!loading&&dataList.length === 0">
    <span class="weui-loadmore__tips"></span>
  </div>
  <div @click="goChat(item)" class="item" v-for="(item) in dataList" :key="item._id">
    <img :src="item.user.avatar" class="avatar">
    <div class="right-content scale-1px">
      <p class="nickname one-line">{{item.user.nickname}}</p>
      <p v-if="item.msg.content && item.msg.content.type==='str'" class="text one-line">{{item.msg.content.value}}</p>
      <p v-if="item.msg.content && item.msg.content.type==='pic'" class="text one-line">[图片]</p>
    </div>
    <div class="time">{{formatTime(item.msg.create)}}</div>
  </div>
</div>
```

每一条的消息列表，采用 flex 布局，样式如下：

```
.item {
  height: 72px;
  width: 100%;
  display: flex;
  position: relative;
  align-items: center;
}
.avatar {
  width: 47px;
  height: 47px;
  border-radius: 5px;
  margin: 15px;
}
.right-content {
  flex: 1;
  margin-left: 15px;
  padding-top: 15px;
  padding-bottom: 11px;
  padding-right: 15px;
  overflow: hidden;
}
```

其中，高度采用固定高度，左边头像的大小采用固定值，右边的数据采用 `flex:1`，即可撑满剩余空间。

另外，我们需要对消息的类型判断文字和图片展示不同的UI：

```
<p v-if="item.msg.content && item.msg.content.type==='str'" class="text one-line">{{item.msg.content.value}}</p>
<p v-if="item.msg.content && item.msg.content.type==='pic'" class="text one-line">[图片]</p>
```

然后，我们需要绑定事件，点击进入聊天页面：

```
  goChat (item) {
    this.$router.push({
      path: '/chat',
      query: {
        id: item.user._id,
        name: item.user.nickname
      }
    })
  },
```

最后，需要编写获取列表数据的逻辑，定义 `fetchData` 方法，代码如下：

```
  async fetchData () {
    this.loading = true
    let resp = await service.get('message/getchatlist', {
      keyword: this.keyword
    })
    this.loading = false
    if (resp.data !== 0) {
      this.dataList = resp.data
    }
  },
```

将 keyword 作为参数传到后端，将获取到的数据赋值给 `this.dataList`。

到此，我们整个聊天列表页的前端 UI 开发已经完成，后面会开发对应的后台接口。

小结

本章节主要讲解了聊天列表页UI开发逻辑。

相关知识点：

1. 在vuex里修改store的状态，采用action和直接修改state有何区别知识点。

本章节完整源代码地址：

Github

}

27 使用 Scoket 改造后端接口

29 聊天列表页面后端接口开发