07 登录页 UI 界面开发

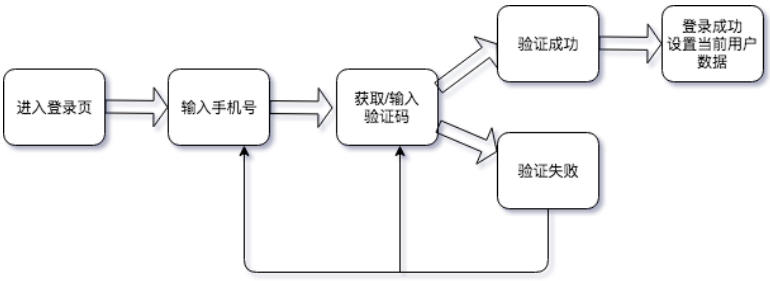
更新时间：2019-07-31 17:37:19

> 生活的理想，就是为了理想的生活。
>
> ——张闻天

各位同学，下面的章节就要进入我们真正的项目页面的逻辑开发了，我们首先从最简单的登录页面开始，相信大家已经准备好了相关的环境，我们这就开始写代码！

登录页面逻辑图：

在开始开发一个页面之前，我们首先需要梳理一下整个页面的逻辑流程，下面这张图就是登录页面的流程：



1. 首先需要输入手机号进行登录。这里我们省去了注册的步骤，而是每次登录采用手机短信验证的方式。
2. 手机号输入成功需要先获取验证码，填上验证码后进行登录。
3. 校验成功后就表明登录成功，然后需要将当前用户的信息设置在vuex的store中。
4. 校验失败则重新尝试，这里我们有限频逻辑，会在后面章节中讲解。

UI效果：

下图是整个页面最终的实现效果：

整个登录页面的UI是相对来说比较简单的，页面上的UI元素很少，下面就进入开发。

引入**weui**

weui 主要由2部分组成，`weui.css` 和 `weui.js`，其中 `weui.css` 主要是提供了常用组件的样式，例如按钮，输入框等等。

而 weui.js 则提供了组件的 JavaScript 调用封装，提供给我们 JavaScript 接口来调用 weui 的组件，例如弹出框，图片上传组件等等。

首先，我们需要安装下载 weui，这里我们采用源码引入的方式：

weui.css下载：下载地址：GitHub
weui.js下载：下载地址：GitHub

我们在前端项目的public文件夹下，新建lib文件夹，存放 weui 的资源文件。

修改 index.html，引入 weui.min.js：

```
<script type="text/javascript" class="lazyload" src="" data-original="<%= BASE_URL %>lib/weui/weui.min.js"></script>
```

在前端项目的入口文件 main.js 中，引入 weui.min.css：

```
import './assets/weui.min.css'
```

这里解释以下为何在 index.html 引入 weui.min.js 而在 main.js 引入 weui.min.css：

- weui.min.js 属于第三方的依赖，我们在index.html引入可以直接使用一个全局变量 window.weui，这样不必在每个页面使用时引入，比较方便，并且我们安装 weui.min.js 也是直接采用下载的方式来引入，并没有放在 node_modules 里来使用。当然，如果各位想通过 node_modules 引入也是完全可以的。
- weui.min.css 也是第三方的样式，我们在 main.js 引入是为了让 webpack 帮助我们进行打包，这样可以让 postcss-px-to-viewport 对 weui.min.css 的单位进行转换。

创建login.vue

在前端项目新建一个 components 文件夹，用来存放一些页面的公共组件，同时新建 login 文件夹，并且新建 index.vue，然后开始编写我们的 UI 了，代码如下：

```
<template>
 <div class="container">
  <div class="close" @click="close"></div>
  <p class="title">手机号登录</p>

  <div class="weui-cell weui-cell_vcode">
    <div class="weui-cell__hd">
      <label class="weui-label">手机号</label>
    </div>
    <div class="weui-cell__bd">
      <input class="weui-input" maxlength="11" type="tel" pattern="^\d{11}$" placeholder="请输入手机号" v-model="phoneNum">
    </div>
    <div class="weui-cell__ft">
      <button v-show="timeCode == 60" class="weui-vcode-btn" @click="getCode">获取验证码</button>
      <div v-show="timeCode != 60" class="time-code weui-vcode-btn">{{timeCode}}s</div>
    </div>
  </div>

  <div class="weui-cell weui-cell_vcode vcode-input scale-1px">
    <div class="weui-cell__hd"><label class="weui-label">验证码</label></div>
    <div class="weui-cell__bd">
      <input v-model="code" class="weui-input" type="number" placeholder="请输入验证码">
    </div>

  </div>
  <a class="weui-btn weui-btn_primary" href="javascript:" @click="signUp">确定</a>

 </div>
</template>
```

上面的UI基本通过weui的方式创建，class 和标签可以直接复制即可，这也是 weui 的方便之处，需要注意的是给手机号的 `<input>`，绑定`v-model="phoneNum"`，可以利用 vue 的双向绑定特性在代码里操作 `this.phoneNum` 即可。后面有很多类似的逻辑，大家要理解好这里。

登录页面主要由两个交互：

1. 登录验证逻辑。
2. 手机验证码获取逻辑。

登录验证逻辑：

```
// 判断手机号和验证码都有值才发送请求
if (this.phoneNum && this.code) {
 //发送登录请求
 let resp = await service.post('users/signup', {
   phoneNum: this.phoneNum,
   code: this.code
 })

 if (resp.code === 0) {
   // 登录成功后，将当前用户的数据存入store，以便后续使用
   this.$store.dispatch('setUser', resp.data)
   // 返回上一页
   this.$router.go(-1)
 }
} else {
 weui.topTips('请输入验证码或手机号码')
}
```

手机验证码获取逻辑：

```
async getCode () {
  // 验证手机号是否合法
  if (!/^\d{11}$/.test(this.phoneNum)) {
    weui.topTips('请输入正确手机号')
    return
  }
  if (this.phoneNum) {
    //发送获取验证码请求
    let resp = await service.post('users/phonecode', {
      phoneNum: this.phoneNum
    })

    if (resp.code === 0) {
      weui.toast('验证码已发送', 1000)
      //动态倒计时
      this.countTimeCode()
    }
  }
},
```

更新倒计时秒数：

```
  this.clearFlag = setInterval(() => {
    //倒计时结束后，重制标志位
    if (this.timeCode == 0) {
      this.timeCode = 60
      clearInterval(this.clearFlag)
      return
    }
    //秒数每次减1
    this.timeCode--
  }, 1000)// 1s调用1次
```

这里需要注意一下，一旦在 vue 里有使用到 `setTimeout` 或者是 `setInterval` 时，要注意在组件销毁时，要清除这些定时器标志位，例如下面这段代码：

```
beforeDestroy () {
  clearInterval(this.clearFlag)
}
```

登录校验拦截器

所谓登录校验拦截器，就是指当前端发起一些 api 请求时，有些请求时需要登录态的，有些并不需要，所以，当前端的登录态过期即 cookie 过期，或者是后端返回失效时，我们需要拿到这个时机，去添加我们的逻辑，借助 axios，可以轻松实现，代码如下：

在前端的项目中新建一个 util 文件下，然后新建一个 service.js，对 axios 做一下2次封装：

```
let service = axios.create({
  baseURL: baseURL,
  withCredentials: true,
  timeout: 30000 // 请求超时时间
})

// 添加respone拦截器
service.interceptors.response.use(
  response => {
    if (response.data.code === 1000) {
      router.push({
        path: 'login',
        name: 'login',
        params: {
        }
      })
      weui.topTips('请先登录')
      //发现登录过期，将本地缓存的用户信息清除
      window.localStorage.removeItem('cuser')
    } else if (response.data.code !== 0) {
      weui.topTips(response.data.msg || '接口请求失败')
    }

    return response.data
  },
  error => {
    return Promise.reject(error.response)
  }
)
```

我们后续的 api 请求，都会调用这个 service.js，同时采用 `interceptors.response`，来给每个请求返回添加拦截器，如果登录态过期，就让用户重新登录即可。

同时，这里还添加了数据错误提示，即如果后端返回一个非正常的错误码，前端页面给出通用的 tips 提示。

最后别忘了补充一下 get 和 post 方法，代码如下：

```
function get (url, params = {}) {

  return service({
    url: url,
    method: 'get',
    headers: {
      'wec-access-token': getCookie('token')
    },
    params
  })
}

// 封装post请求
function post (url, data = {}) {
  // 默认配置
  let sendObject = {
    url: url,
    method: 'post',
    headers: {
      'Content-Type': 'application/json;charset=UTF-8',
      'wec-access-token': getCookie('token')
    },
    data: data
  }
  return service(sendObject)
}
```

这里我们采用将后端返回的 token 放在 header 里面，来防止 CSRF，我们会在后面的章节具体讲解。

小结

本章主要讲解了一些通用样式和组件的引入，和登录页面相关的逻辑，整个登录页面的UI相对比较简单。
相关知识点如下：

1. 使用 v-model 结合 `<input>` 来实现 vue 中双向绑定。
2. 一旦在vue里有使用到 `setTimeout` 或者是 `setInterval` 时，要注意在组件销毁时，要清除这些定时器。

}

06 初始化后端项目

vuex 和 async/await 使用说明