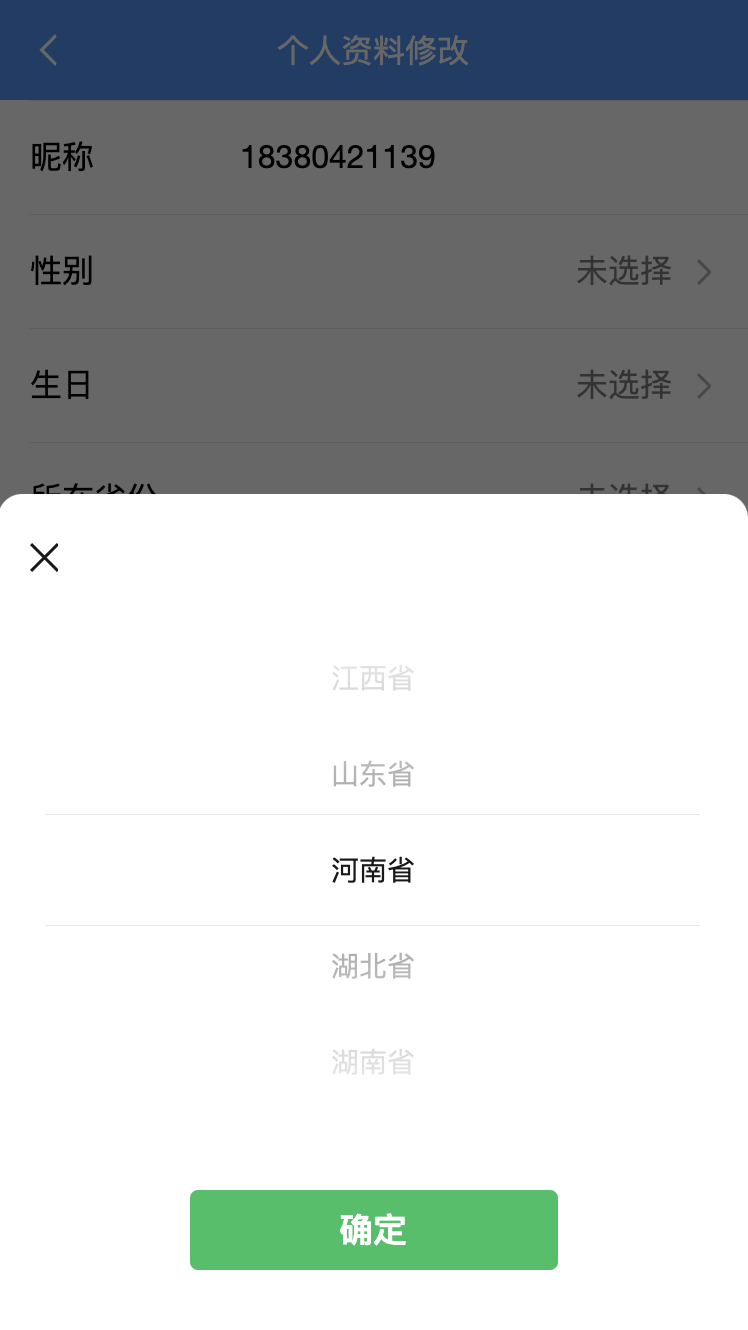
**源码健身Fitness移动Web站**

**个人资料修改Day04**

# 今日任务概览

**完成页面：个人资料修改（静态布局 + 接口联调）**



附：

设计稿地址：[设计稿链接](https://modao.cc/app/629ff0afd82a66bebb7e0d87b726d618140faf20?simulator_type=device&sticky)

# 修炼技术点

* 图片文件上传、FormData
* 移动Web基础样式库WeUI
* 省市二级联动
* 表单处理
* Ajax请求接口联调、PostMan的使用

# 编辑上传头像文件

## 获取本地文件

1.首先，创建一个type类型是file的<input>元素：

<input type="file" id="userImgIpt">

2.通过它，我们本地选择文件后，文件名会自动显示出来：



3.但是我们需要将选择的文件上传服务器，所以要获取本地选择的文件。为<input>元素绑定change事件处理函数，这样本地选择文件后就会触发change事件，并通过this.files拿到本地选择的文件列表：

document.querySelector('#userImgIpt').addEventListener('change', function() {

const fileList = this.files; // 得到上传的文件列表

const file = fileList[0]; // 得到文件对象

console.log('>>>>> fileList', fileList)

})

注意，如果为<input>设置multiple属性，则支持一次性选择多个文件。

获得的**File对象**提供了3个十分有用的属性：

1. name：文件名称
2. size：文件大小，字节数
3. type：文件MIME类型

## 上传文件到服务器

先写出使用XMLHTTPRequest（XHR）发请求的一般流程：

const url = 'http://139.159.146.104:8088/users/upload';

const xhr = new XMLHttpRequest();

xhr.open('POST', url);

xhr.onreadystatechange = function() {

if (xhr.readyState === 4 && xhr.status === 200) {

console.log('响应成功！', JSON.parse(xhr.responseText));

}

}

xhr.send();

关键是最后一步，xhr.send()，send什么？这里需要将文件对象发送到服务器。

使用XMLHttpRequest有2种方式请求数据：

1. 使用AJAX
2. 使用FormData API：[FormData | MDN](https://developer.mozilla.org/zh-CN/docs/Web/API/FormData)

FormData提供了表示表单数据的**键值对（key/value）**的构造方式，可以将数据通过XMLHttpRequest.send()方法发送出去

FormData的用法：

1. 创建一个FormData对象：

const fd = new FormData();

1. 添加新的属性值，append()方法：

fd.append('imgurl', file);

1. 获取key对应的value，get()方法：

fd.get('imgurl');

构造完成FormData的数据后，通过XMLHttpRequest.send()方法上传即可：

完整代码：

const url = 'http://139.159.146.104:8088/users/upload';

const xhr = new XMLHttpRequest();

const fd = new FormData();

fd.append('imgurl', file);

xhr.open('POST', url);

xhr.onreadystatechange = function() {

if (xhr.readyState === 4 && xhr.status === 200) {

console.log('响应成功！', JSON.parse(xhr.responseText));

}

}

xhr.send(fd);

## 使用自定义的文件上传按钮

一个经典用法是——通过click()方法使用隐藏的file input元素：

1. 给input元素添加display: none样式，来隐藏掉公认难看的file input元素
2. 当我们自定义的元素被点击时，调用input元素的click()方法来触发文件上传

另外项目中头像区域的点击，需要使用e.perventDefault()来组织默认的点击跳转事件。

const userImgIpt = document.querySelector('#userImgIpt')

document.querySelector('#userImg').addEventListener('click', function(*e*) {

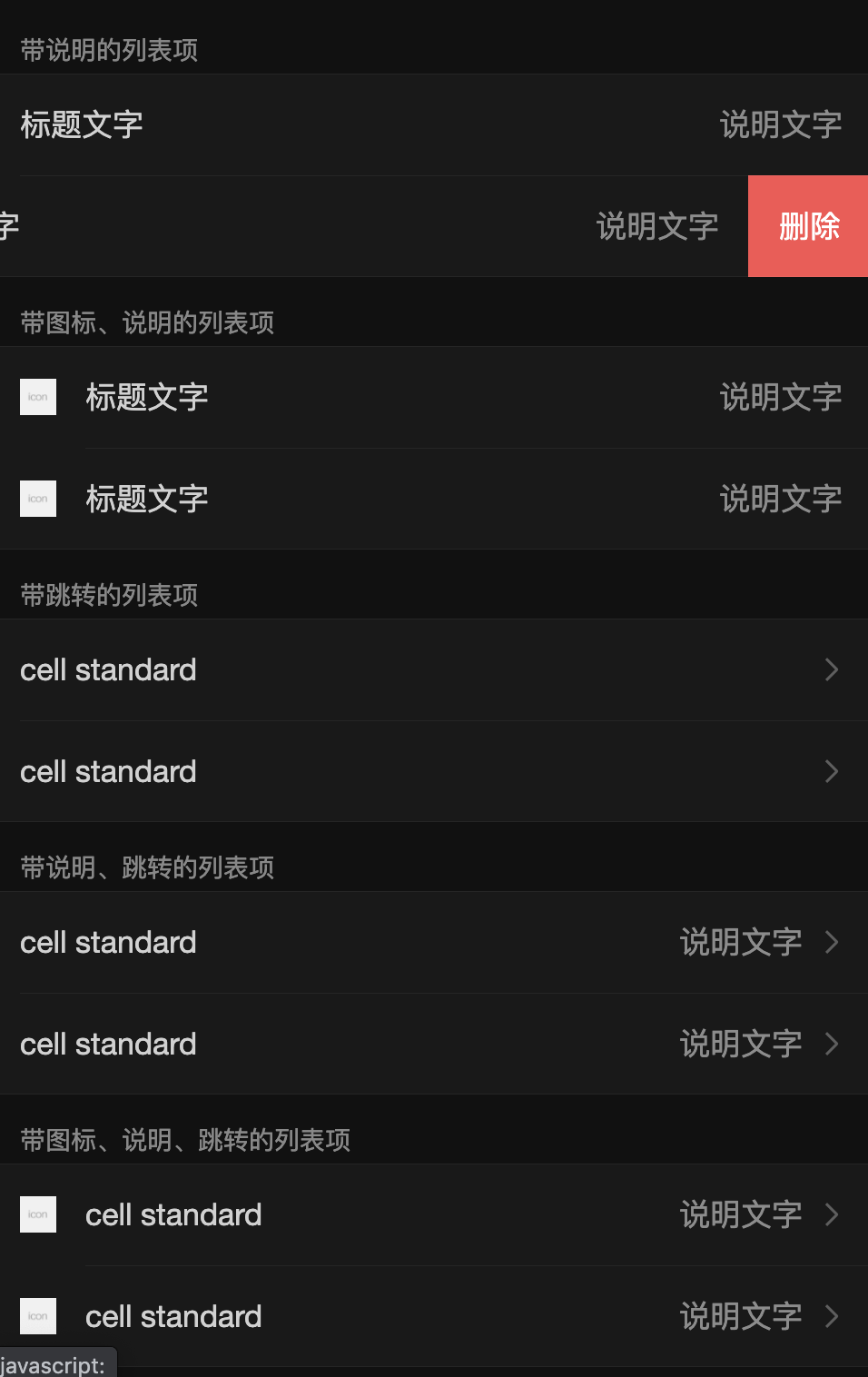
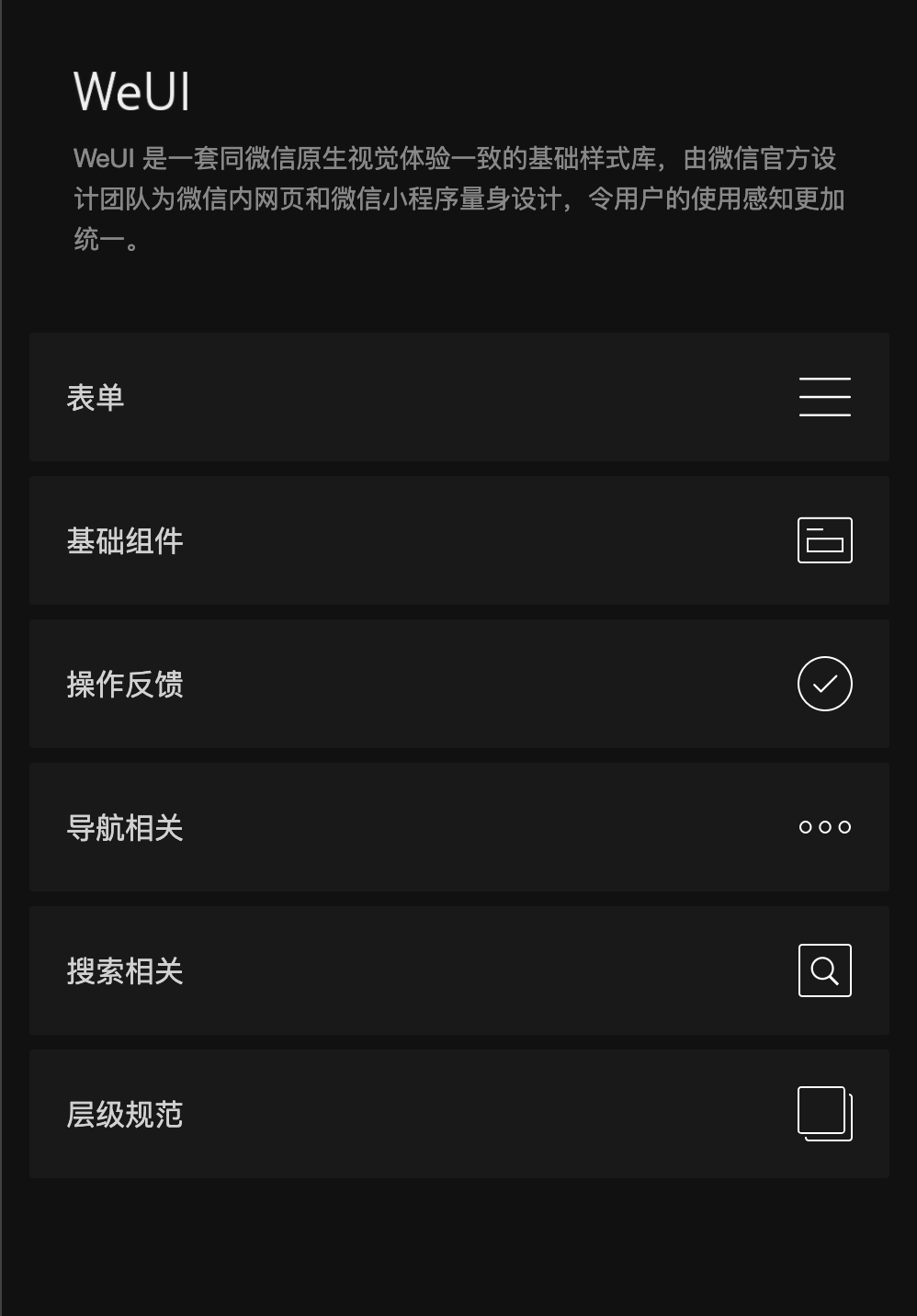
e.preventDefault() // 阻止事件默认操作

userImgIpt.click() // 触发 file input

})

# 移动Web基础样式库WeUI

## WeUI的引入



WeUI 是一套同微信原生视觉体验一致的基础样式库，由微信官方设计团队为微信内网页和微信小程序量身设计，令用户的使用感知更加统一。

* 官网示例：<https://weui.io/>
* GitHub：<https://github.com/Tencent/weui>
* WeUI说明文档Wiki：<https://github.com/Tencent/weui/wiki>

为什么要使用它？

* 它提供了许多移动Web常用基础布局控件，如：按钮、表单、弹窗、选择器、列表等等。可以大大节省布局阶段的工作量，使得我们更加专注地处理JS逻辑；
* 微信官方团队研发，与微信的设计规范一致，更加注重用户体验
* 良好的浏览器兼容性；
* 即便不直接拿来使用，也可以作为代码参考

引入WeUI：

参考 [快速上手](https://github.com/Tencent/weui/wiki/getting-started) 文档，引入WeUI的CSS、JS的CDN即可（推荐用CDN，下载本地有编码问题）：

<link rel="stylesheet" href="https://res.wx.qq.com/open/libs/weui/2.3.0/weui.min.css"/>

<script src="https://res.wx.qq.com/open/libs/weuijs/1.2.1/weui.min.js" charset="utf-8"></script>

## 模拟选择器Picker的使用

Picker使用效果示例：<https://weui.io/#picker>

Picker的调用代码：[代码](https://github.com/Tencent/weui/blob/master/src/example/picker/picker.html)

阅读代码我们可以发现，真正调出选择器蒙层的，是调用weui.picker()，至于触发元素是怎么样的，并没有做限制，因此我们可以自定义触发选择器蒙层弹出的元素。

另外对于weui.picker()的传参，第一个参数是一个配置数组，数组的每个元素是配置对象：

[{

label: '男', // 要显示的文字

value: '男' // 选择后的值

}, {

label: '女',

value: '女'

}]

也就是说，调用这类控件的关键是写好配置项。

第二个参数是个对象，比较关键的是配置当用户选择确认时的回调onConfirm：

{

onConfirm: function (*res*) { /\* 确认时触发 \*/

sexDisplay.textContent = res[0].value;

},

title: '性别选择'

}

由此，第三方交互控件的调用，重点在得到用于配置的配置项，另外即便说明文档不充分，我们也可以通过命名揣测、穿插console.log调试等方式确定它们的用途。

# 省市二级联动

对于较复杂的场景，一定要先分析，再动手编码。先思考出流程方案来，甚至写出伪代码，然后再补全流程进行实际编码。

项目中省市联动的重难点，在于：

1. 较多的DOM操作，无论是获取元素绑定事件，还是拿到数据修改元素的显示
2. 数据的整合：拿到接口数据整合成WeUI需要的配置项
3. 函数嵌套着函数：会遇到回调函数中又执行操作嵌套回调函数
4. DOM元素事件的解绑，使用removeEventListener

省、市联动方案分析：

// 省、市联动方案分析

// 1.请求省份数据

// 2.构造出用于 WeUI Picker 使用的配置项数组（注意 value 是省份 Id，后面用于请求城市）

// 3.监听省份点击，调 weui.picker()

// 4.confirm：

// (1) 更改省的显示

// (2) 将城市选择重置

// (3) 用选择的省 ID 请求城市接口

// 5.构造城市配置项 -> 绑定点击事件 -> 调 weui.picker()

// 6.难点：每次绑定城市选择事件时，得清空旧的已绑定的事件！

完整代码实现：

let preventHandleCityPick = null

window.$http.get('/address/province', function(*pData*) {

if (pData.status === 0) {

// 构建省份选择项

const provinceOptions = pData.data.map(function(*item*) {

return {

label: item.name,

value: item.addressId

}

});

// 处理省份选择

provincePicker.addEventListener('click', function() {

weui.picker(provinceOptions, {

onConfirm: function(*res*) {

provinceDisplay.textContent = res[0].label // 改变省的显示

cityDisplay.textContent = '未选择'; // 将市选择重置

const selectedProvinceId = res[0].value // 选择的省 ID

// 构建市选择项

const requestCityUrl = '/address/city/' + selectedProvinceId

window.$http.get(requestCityUrl, function(*cData*) {

if (cData.status === 0) {

const cityOptions = cData.data.map(function(*item*) {

return {

label: item.name,

value: item.addressId

}

});

// 处理城市选择

function handlePickCity() {

weui.picker(cityOptions, {

onConfirm: function(*res2*) {

cityDisplay.textContent = res2[0].label; // 改变市的显示

}

});

}

// 清空已绑定的市选择事件处理

cityPicker.removeEventListener('click', preventHandleCityPick);

// 处理市选择

preventHandleCityPick = handlePickCity

cityPicker.addEventListener('click', handlePickCity);

}

})

}

});

});

}

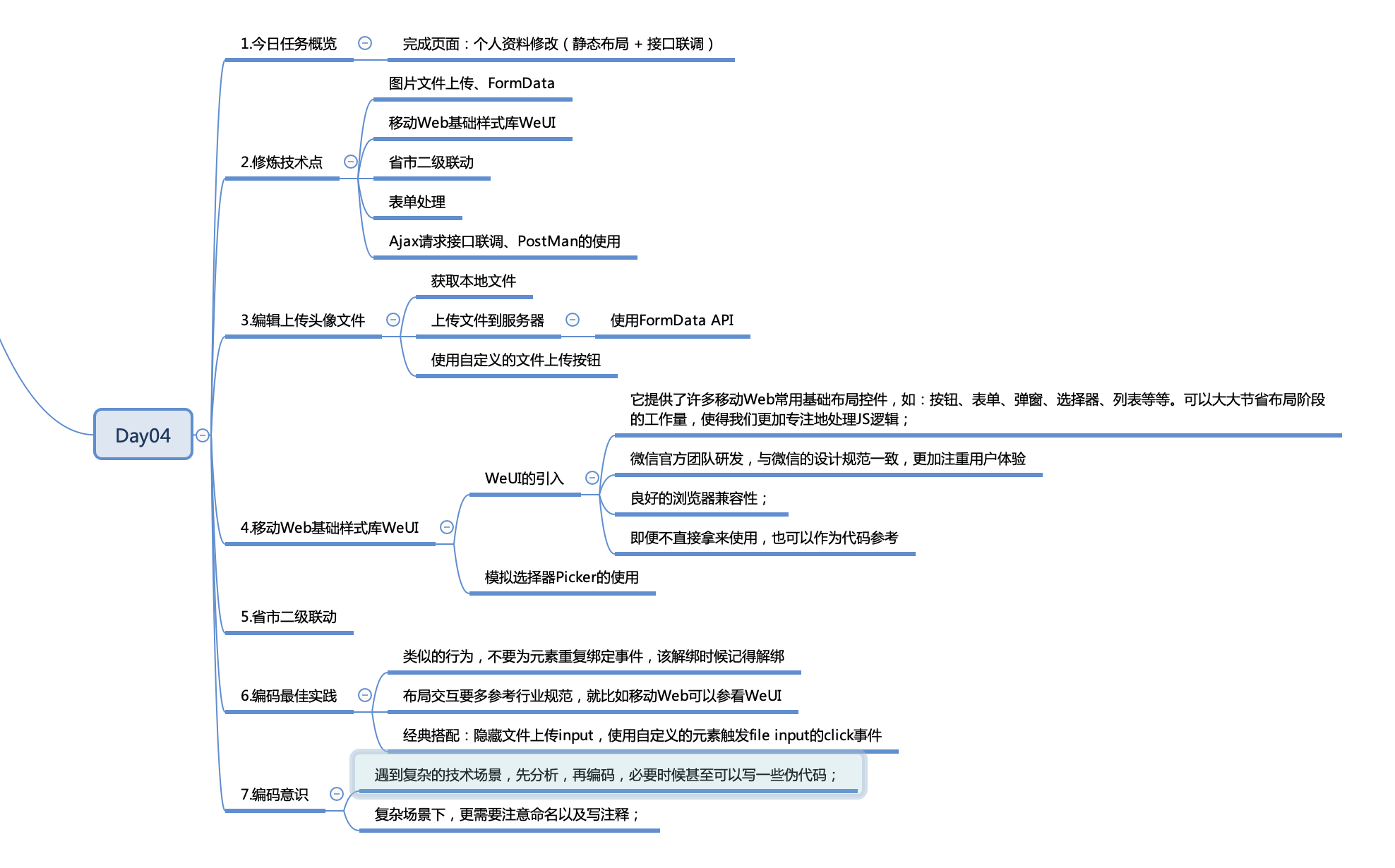
});

# 今日研发任务CheckList

* 完成头像上传逻辑，并封装$updateFile函数，放在http.js中；
* 使用WeUI完成个人资料修改页的静态布局
* 完成省、市二级联动逻辑
* 完成表单验证逻辑

PS：使用Git进行代码版本控制，commit格式：git commit -m “feat: 完成注册页背景图适配”

# 今日小结



## 编码最佳实践

* 类似的行为，不要为元素重复绑定事件，该解绑时候记得解绑
* 布局交互要多参考行业规范，就比如移动Web可以参看WeUI
* 经典搭配：隐藏文件上传input，使用自定义的元素触发file input的click事件

## 编码意识

* 遇到复杂的技术场景，先分析，再编码，必要时候甚至可以写一些伪代码；
* 复杂场景下，更需要注意命名以及写注释；