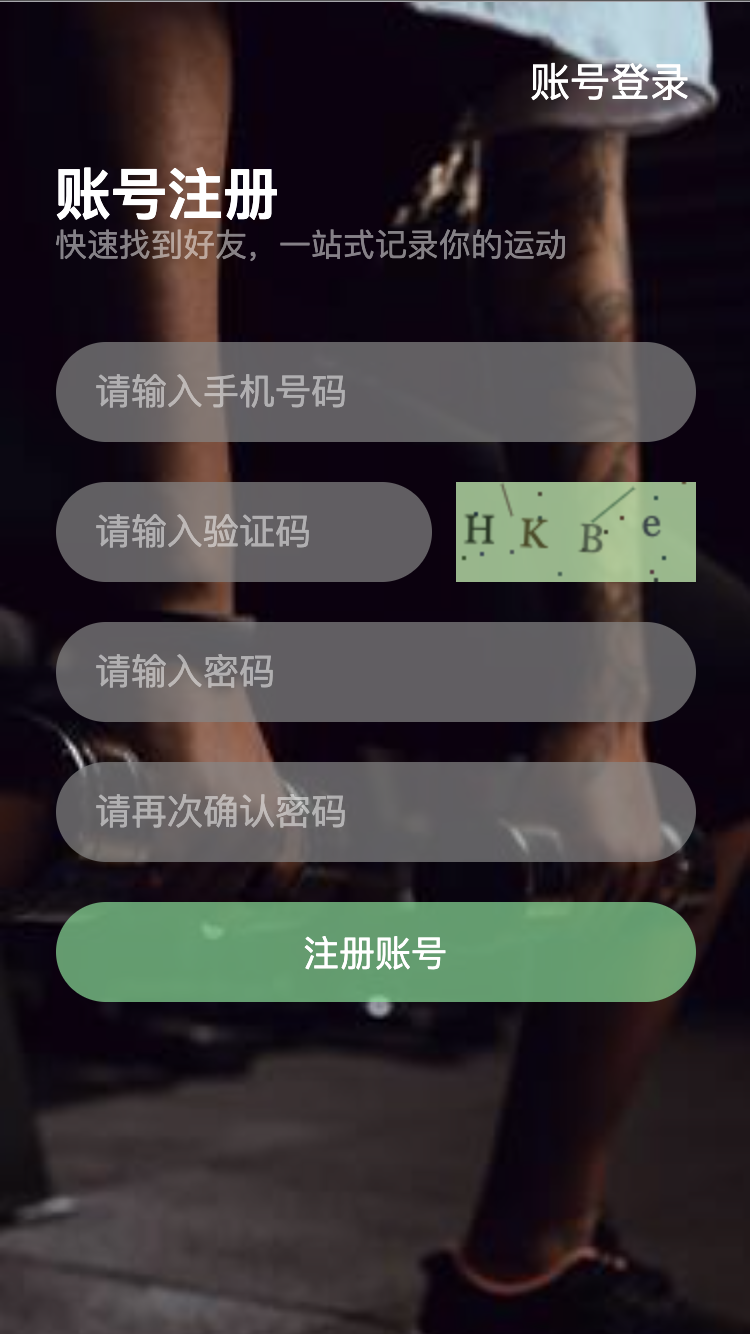
**源码健身Fitness移动Web站**

**广告&注册&登录Day02**

# 今日任务概览

**完成页面：广告 / 注册 / 登录（静态布局 + 接口联调）**

广告 注册 登录



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

* 移动端布局适配
* CSS3动画 @keyframes
* 间隔定时器 window.setInterval() 与 window.clearInterval()
* 使用JS生成验证码的第三方库CaptchaMini
* 表单验证
* 本地存储
* AJAX请求、PostMan的使用

# 广告页开发

## 背景图在不同机型尺寸下的适配

适配策略：

1. 背景图的宽、高都要比手机屏幕的显示区域大
2. 设置背景图的位置：水平居中、顶部对齐

关键代码：

/\* 广告图区域 \*/

.ad-container {

flex: 1;

background-image: url(../image/advertisement-bg.jpg);

background-size: 1280px;

background-position: top center;

background-repeat: no-repeat;

}

## 标语由上至下淡入的动画效果

技术点：CSS3关键帧 @keyframes

实现：先定义动画效果 @keyframes fadeup，再在CSS选择器中使用它 animation-name: fadeup

关键代码：

/\* 标语 \*/

.ad-slogen {

...

animation-name: fadeup;

animation-duration: 3s;

}

// 由上至下淡入

@keyframes fadeup {

from {

opacity: 0;

bottom: 164px;

}

to {

opacity: 1;

bottom: 124px;

}

}

## 倒计时跳转

技术点：设置与清空定时器window.setInterval()、window.clearInterval()；页面跳转window.location.href

HTML：

<div id="adBtn">

<span id="adCount">5</span>s | 跳过

</div>

JavaScript：

/\*\*

\* 广告页

\*/

document.ready(function() {

const adCount = document.querySelector('#adCount'); // 取 DOM 元素

// 倒计时

const intervalId = window.setInterval(function() {

// Node.textContent: https://developer.mozilla.org/zh-CN/docs/Web/API/Node/textContent

const toSetCount = adCount.textContent - 1;

adCount.textContent = toSetCount

if (toSetCount === 0) {

// 停止循环计时

window.clearInterval(intervalId);

// 进行页面跳转

window.location.href = './login.html';

}

}, 1000);

// 跳过按钮

const adBtn = document.querySelector('#adBtn');

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

window.location.href = './login.html';

});

});

# 注册页开发

## 可复用的表单控件样式

技术点：

1. 把这种公共的布局样式，抽到less/common/basic.css中；
2. 改变表单元素的占位文本样式（了解）：使用伪元素placeholder：<https://developer.mozilla.org/zh-CN/docs/Web/CSS/::placeholder>

CSS：

/\* 注册/登录 文本输入框 \*/

.ipt-lg {

padding-left: 20px;

width: 100%;

height: 50px;

background-color: rgba(134, 134, 134, 0.7);

color: #fff;

font-size: 18px;

line-height: 50px;

border-radius: 25px;

letter-spacing: 1px;

}

.ipt-lg::placeholder { /\* 伪元素 placeholder \*/

letter-spacing: 0;

color: rgba(255, 255, 255, 0.5)

}

## 使用第三方库生成验证码（了解）

第三方库captcha-mini：<https://github.com/saucxs/captcha-mini>

重点在于理解使用一个第三方库的流程：

1. 引入资源：<link>引CSS，<script>引JS
2. 确保HTML中有Canvas元素，并设置width、height
3. JS中调用第三方库的API，进行渲染、取到渲染后的验证码字符串，以备后续的表单提交时验证

HTML：

<!-- 验证码 Canvas -->

<canvas id="captcha" class="ml12" width="120" height="50"></canvas>

JavaScript：

let captchaCode = ''; // 当前验证码

// 创建验证码渲染器

const captchaDrawer = new CaptchaMini();

captchaDrawer.draw(document.querySelector('#captcha'), function(*code*) {

captchaCode = code;

});

## 注册流程解析

点击注册按钮后，要进行以下流程处理：

1. 表单验证：判空、手机号格式、验证码、密码一致性
2. 请求注册接口
3. 注册成功后自动登录，跳转到首页

### 表单验证

先获取表单元素，并缓存住：

// 获取表单元素

const phoneNumberIpt = document.querySelector('#phoneNumberIpt'); // 手机号输入框

const captchaIpt = document.querySelector('#captchaIpt'); // 验证码输入框

const pwdIpt = document.querySelector('#pwdIpt'); // 密码输入框

const pwdAgainIpt = document.querySelector('#pwdAgainIpt'); // 再次确认密码输入框

const hint = document.querySelector('#hint'); // 表单验证提示

点击注册按钮后，对表单元素取值，然后写验证逻辑，过程中不满足则return：

// 对表单元素取值

const phoneNumberValue = phoneNumberIpt.value

const captchaValue = captchaIpt.value

const pwdValue = pwdIpt.value

const pwdAgainValue = pwdAgainIpt.value

// 验证手机号

if (phoneNumberValue === '') {

hint.textContent = '手机号不能为空！'

return;

}

else if (!window.utils.isPhoneNumber(phoneNumberValue)) {

hint.textContent = '手机号格式错误！'

return;

}

// 验证验证码

else if (captchaValue === '') {

hint.textContent = '验证码不能为空！'

return;

}

else if (captchaValue.toLowerCase() !== captchaCode.toLowerCase()) { // 注意切小写

hint.textContent = '验证码错误！'

return;

}

// 密码

else if (pwdValue === '') {

hint.textContent = '密码不能为空！'

return;

}

else if (pwdValue.length > 20) {

hint.textContent = '密码过长，请确保长度小于20字符！'

return;

}

else if (pwdAgainValue !== pwdIpt.value) {

hint.textContent = '密码不一致！'

return;

}

// 请求注册接口

// ...

### 验证手机号格式-正则的使用

正则表达式的使用：<https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Guide/Regular_Expressions>

utils中写一个验证手机号格式的工具函数：

// 验证是否是手机号

function isPhoneNumber(*number*) {

return /^1[3-9]\d{9}$/.test(number);

}

window.utils = {

isPhoneNumber: isPhoneNumber

}

然后在业务代码中调用：

else if (!window.utils.isPhoneNumber(phoneNumberValue)) {

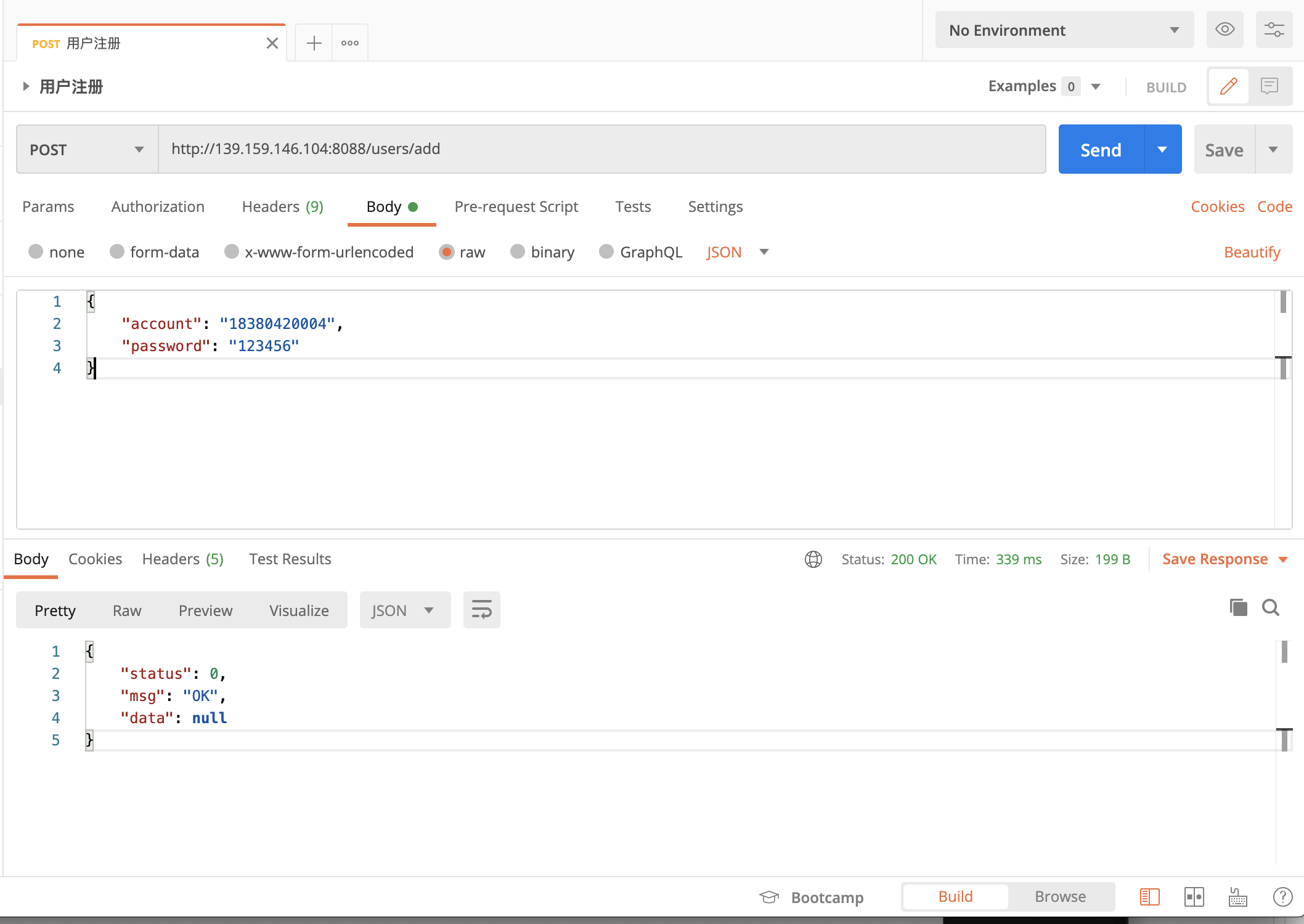
hint.textContent = '手机号格式错误！'

return;

}

### 请求注册接口

先用PostMan进行接口测试，观察返回数据：



接口请求的编码（统一status返回为0时表示成功）：

// 请求注册接口

window.$http.post('/users/add', {

account: phoneNumberValue,

password: pwdValue

}, function(*res1*) {

// 注册成功

if (res1.status === 0) {

hint.textContent = '注册成功！！';

// 注册成功后自动登录，并跳转

// ...

}

});

# 登录页开发

登录后将用户数据保存在本地存储LocalStorage中，以便后续其他页面使用（尤其是userId）：

// 请求登录接口

window.$http.post('/users/login', {

account: phoneNumberValue,

password: pwdValue

}, function(*res*) {

// 登录成功

if (res.status === 0) {

const userData = res.data.user

// 将用户数据存储在本地存储中

window.localStorage.setItem('user', JSON.stringify({

userId: userData.userId, // 用户 ID

account: userData.userData, // 账号

nickname: userData.nickname, // 昵称

gender: userData.gender, // 性别

birthday: userData.birthday, // 生日

imgurl: userData.imgurl, // 头像图片地址

address: userData.address, // 省市区地址

sign: userData.sign // 个人签名

}));

window.localStorage.setItem('token', res.data.token);

// URL 跳转至首页

window.location.href = './index.html';

} else {

// 登录失败

hint.textContent = res.msg;

}

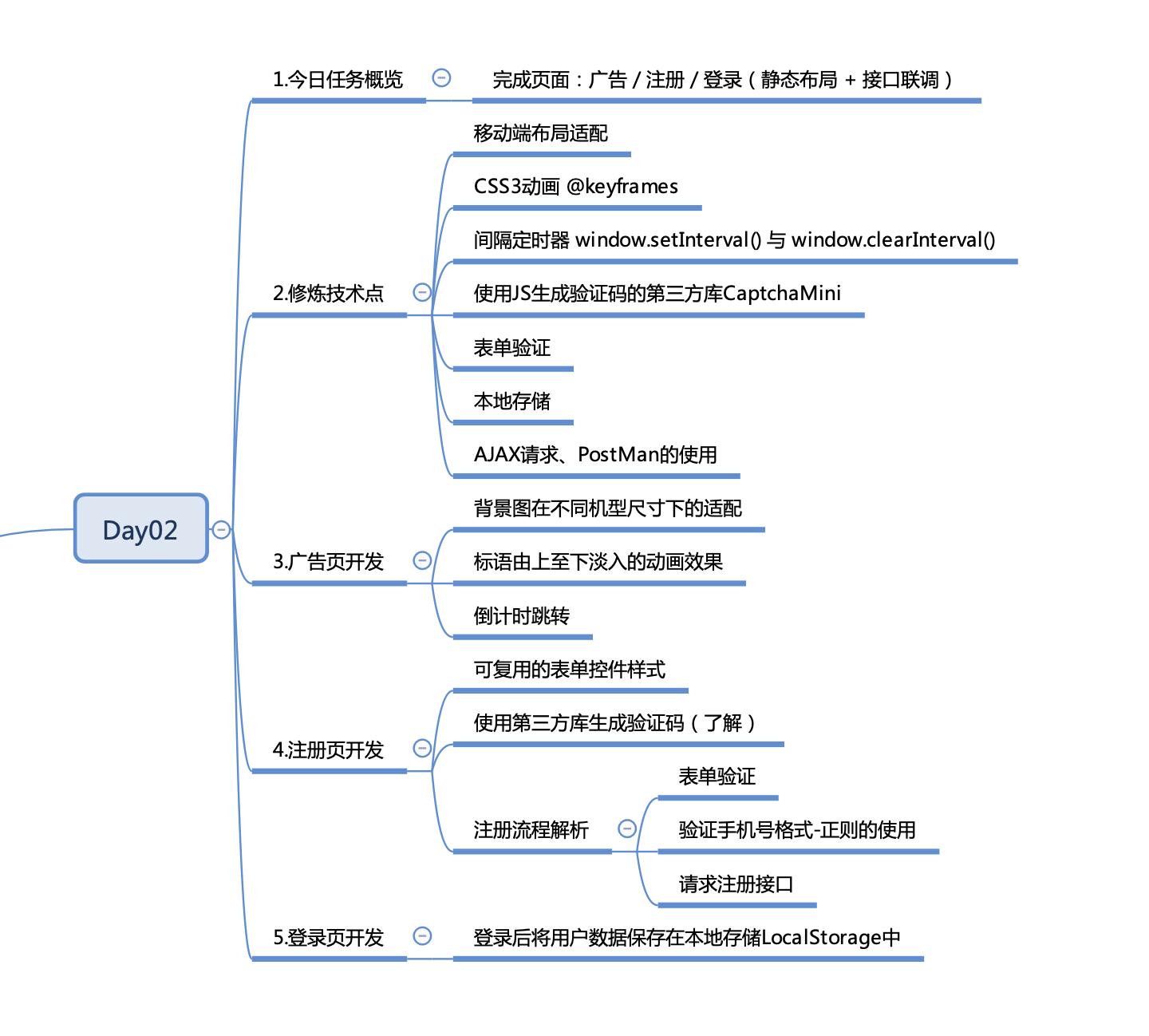
})

# 今日研发任务CheckList

* 广告页：按授课，完成自己广告页实现的优化
* 注册页：
  + 背景图适配
  + 右上角跳转至登录
  + 表单控件样式的复用（含提交按钮样式复用）
  + 验证码渲染（Flex布局）+验证
  + 提交前的表单验证 + 表单验证的出错提示
  + 注册接口联调
  + 可选：注册成功后，自动登录，跳转首页
* 登录页：登录后数据入本地存储LocalStorage，页面自动跳转至首页

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

* 使用封装的document.ready()，原理是监听DOMContentLoaded事件，DOM加载完便执行JS，而无需等待所有媒体资源的加载；
* 涉及DOM元素查询的场景，先用变量缓存已查询的DOM元素，以备后续使用。尤其在定时器，或是其他可能多次调用的函数中，避免不必要的重新查询DOM；
* 移动端背景图适配：背景图的宽、高都要比手机屏幕的显示区域大，水平居中，顶部对齐

## 编码意识

* 可复用：抽出可以复用的CSS、JS代码
* 性能意识：遇到可能多次执行的函数，一定要考虑它的内部实现是否有纰漏，造成无谓的性能的损耗
* 可读性：window.setInterval()、window.utils.isPhoneNumber() 等，前面加window.前缀，表明是一个BOM方法，或者是工具函数，与页面JS中的一般函数做区分，一目了然；
* 正则的使用：了解正则语法（可查API文档）+ 搜索正则表达式 = 可以用正则搞定功能！
* 第三方库的使用：找到API文档，引入资源，建立HTML元素，JS调API
* 接口联调：先PostMan上测，测通了再写请求代码！