app.js

· **onLaunch()**: 这是应用的启动时触发的函数。可以用于初始化应用状态和处理一些启动时需要执行的逻辑。

· **wx.getStorageSync()**: 用于同步获取本地存储的数据。在这里用来获取日志。

· **wx.setStorageSync()**: 用于同步设置本地存储的数据。这里将最新的日志存储起来。

· **wx.login()**: 调用微信的登录接口，用于获取用户的登录凭证（code），通常需要进一步向服务器请求用户信息。

account.js

· **数据管理**: 使用 data 属性来管理页面的状态，包括支出限额和账户余额。

· **生命周期管理**: 在 onLoad() 函数中调用 loadAccountData() 来加载用户的设置。

· **数据存储**: 使用 wx.getStorageSync 和 wx.setStorageSync 来进行本地存储操作，使得用户的设置在应用重启后依然有效。

· **用户交互**:

* onInputLimit 函数处理用户在输入框中的输入。
* saveLimit 函数保存用户的支出限额，并给出提示。
* updateBalance 函数用于更新账户余额，可以通过其他业务逻辑调用。

index.js

· **数据管理**: 使用 data 属性来存储记账记录，初始化为空数组。

· **生命周期管理**: 在 onLoad() 函数中调用 loadRecords() 来加载用户的记账记录。

· **数据存储**: 使用 wx.getStorageSync 获取本地存储中的记录，确保用户的数据持久化。

· **导航功能**:

* goToRecord() 函数用于导航到记账记录页面。
* goToStatistics() 函数用于导航到统计页面。
* goToAccount() 函数用于导航到账户管理页面。

logs.js

· **数据管理**: 使用 data 属性来存储格式化后的日志记录，初始化为空数组。

· **工具函数**: 引入工具函数 util.js，用于格式化时间，确保日志记录以易读的形式展示。

· **生命周期管理**: 在 onLoad() 函数中获取本地存储的日志数据，并将其格式化后存入 logs 属性。

record.js

· **数据管理**: 使用 data 属性存储用户输入的日期、金额、类别、备注和记录类型。

· **用户输入处理**: 通过多个处理函数管理用户输入的日期、金额、备注和类别。

· **金额验证**: 在 saveRecord 函数中验证输入金额是否有效，确保它是一个正数。

· **收入与支出处理**: 根据记录类型（收入或支出）更新账户余额，并处理余额不足的情况。

· **日期格式化**: 提供一个函数将输入的日期格式化为 YYYY-MM-DD 格式。

· **本地存储更新**: 将新记录存储到本地，并在保存后返回上一页。

static.js

· **数据管理**: 使用 data 属性存储总收入、总支出、选择的类别和日期等信息。

· **统计计算**: 在 calculateStatistics 函数中，通过遍历所有记录来计算总收入和支出，同时根据用户选择的日期和类别进行筛选。

· **用户输入处理**: 处理用户选择的类别和日期，通过事件函数更新状态并重新计算统计数据。

· **日期确认**: 提供 confirmDate 方法，用于确认用户选择的日期，并更新数据显示。