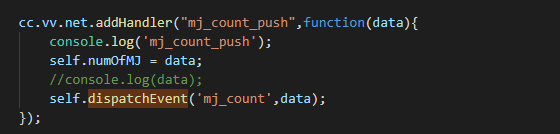
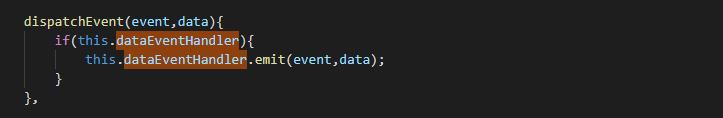
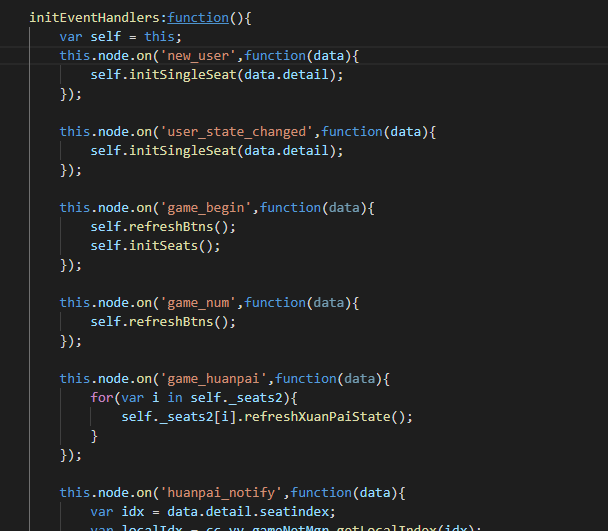
## 项目大致框架流程

1. Loading.fire 启动界面，执行脚本loadingLogic
2. LoadingLogic:初始化各种管理器和把静态类保存在cc.vv数组中，利用http请求检测版本号(缺资源更新部分)，发现最新版本，则加载资源，跳转login场景
3. login部分（游客登录流程）:游客向账号服发送HTTP游客请求，账号服收到后返回客户端账号、sid、大厅服地址。客户端接收到数据后，修改自定义HTTP对象的url地址为大厅服地址，然后向大厅服发送HTTP登录请求(到目前为止，根据我观察，在进入游戏服前，不会创建长连接，大厅部分获取数据也只会通过HTTP请求获得，好处当然是省掉长连接的消耗，坏处就是只能单方请求更新，不能服务器主动推送)
4. userMgr.js:保存了大多数用户数据和大厅的消息数据，当显示界面需要数据显示时就从该管理器获取数据，还封装了关于用户的认证，登录等请求函数
5. Http.js:封装了HTTP的请求接口的管理器
6. Net.js:封装了socket.io的各种接口函数的管理器
7. GameNetMgr.js:封装了游戏网络协议处理接口，并负责把数据派发到麻将场景进行处理，相关的派发流程如下

把majiang node赋值到gameNetMgr中保存

然后收到网络消息后，可以对node进行相应的时间派发(GameNetMgr.js)场景初始化时进行majiang node 事件的注册(MJRoom.js)



## 热更新部分:

version.manifest和project.manifest的格式本质上是一致的，version.manifest中只包含大的版本号信息，而project.manifest中包含version.manifest中所有内容+所有项目文件信息，这样做的好处是当项目文件很多时project.manifest会比较大，所以单独分割出来一个version.manifest来比较大的版本，如果大版本一致就不用下载project.manifest了