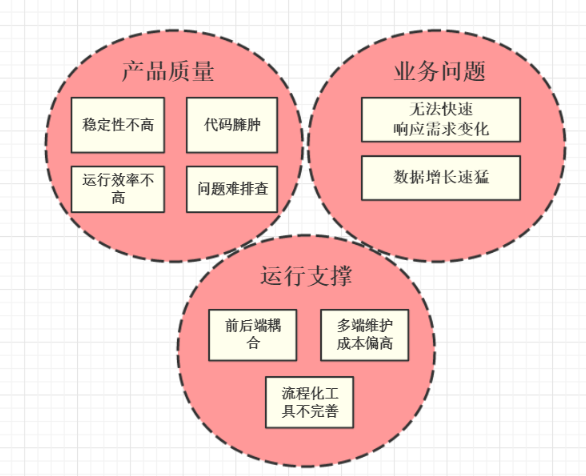
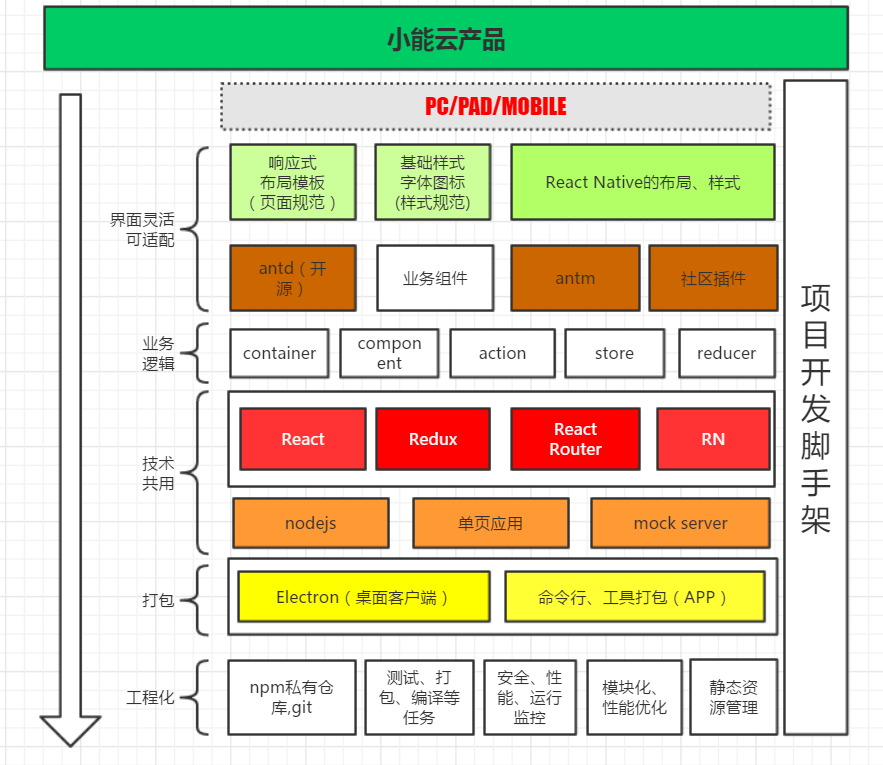
## 客户端重构背景



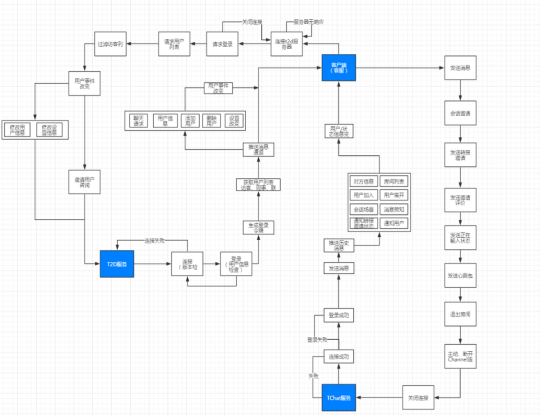
## 重构原则

1. 前端开发脚手架 + 开源产品组合 + 少量插件开发
2. 移动优先原则
3. SPA应用

## 里程碑

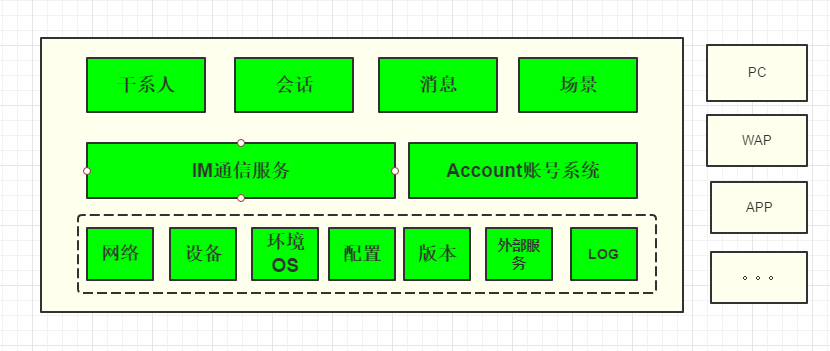


## 原咨询接待业务流程：（TChat<—>客户端<—>T2D）

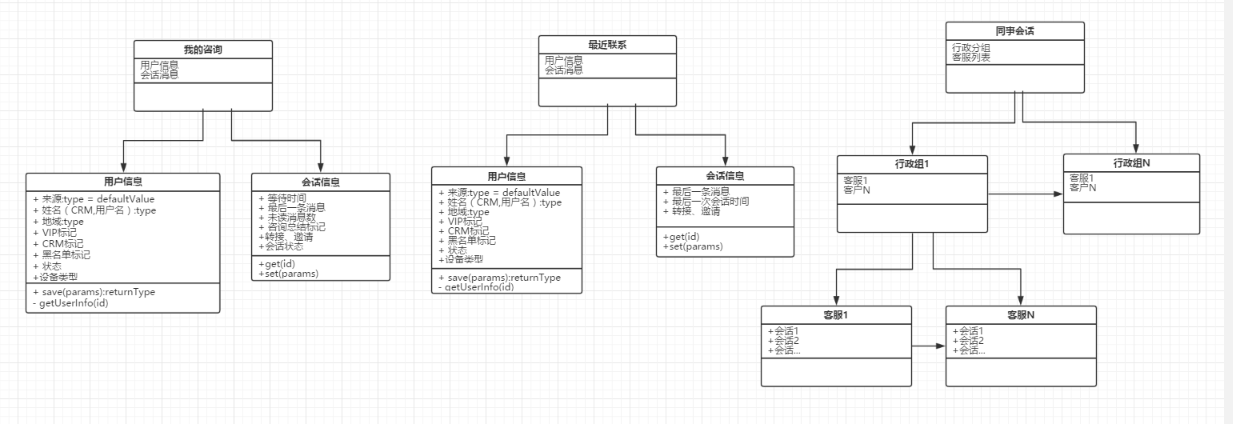


## 基本类库结构设计

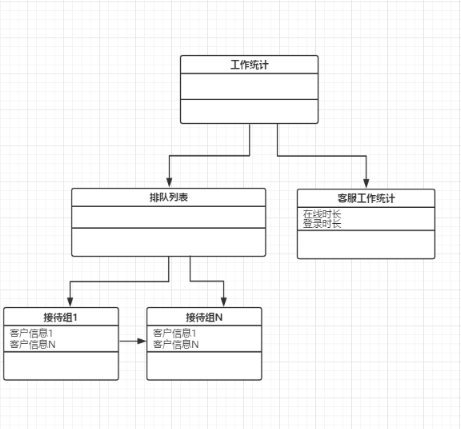
## 待客云架构设计（业务模型设计+基础类库）



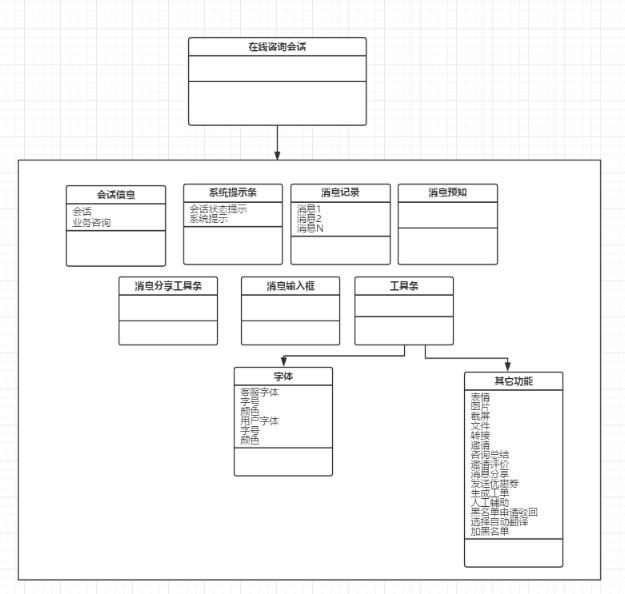
### 用户列表



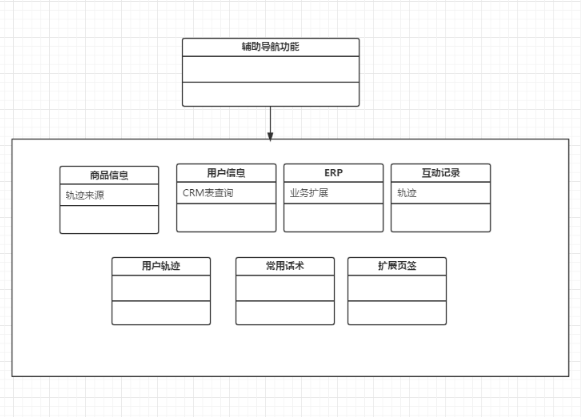
### 工作统计



### 在线咨询会话

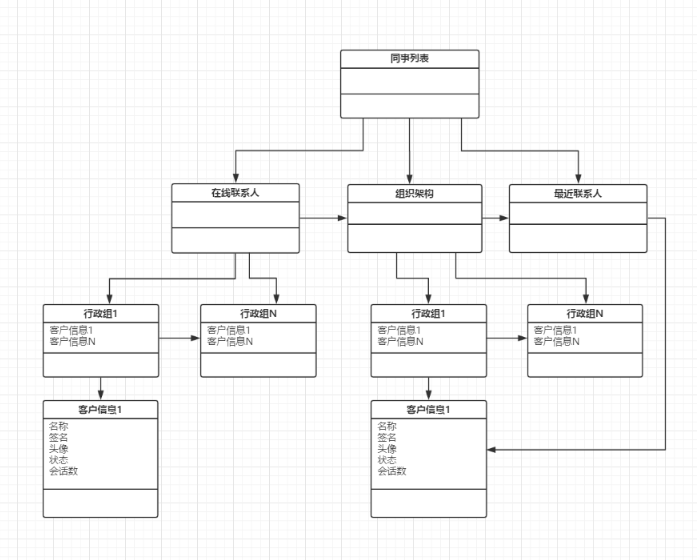


### 辅助导航栏

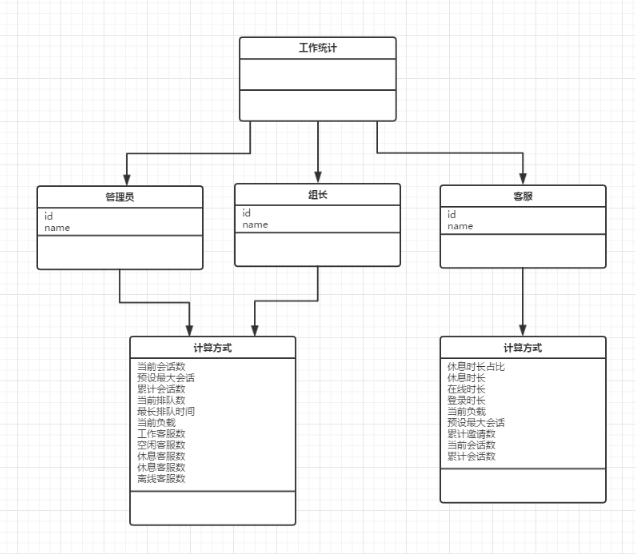


## 我的同事（业务模型设计草图）

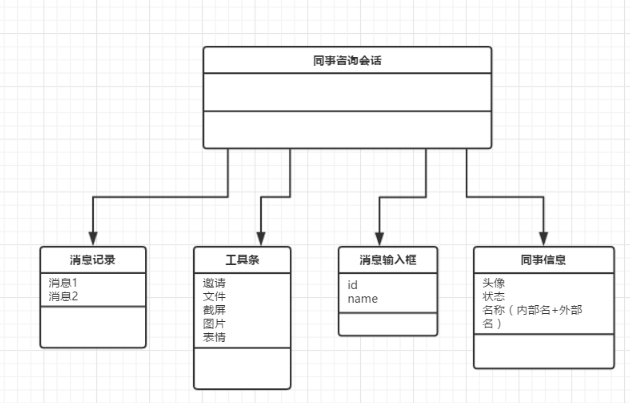
### 同事列表



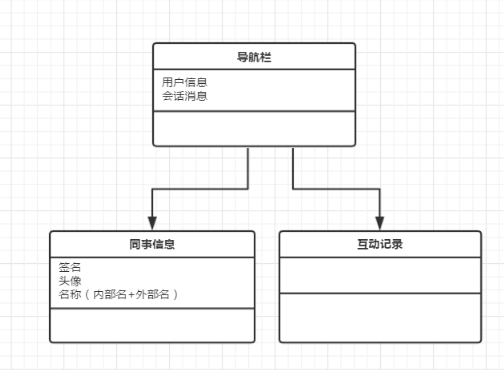
### 工作统计



### 同事咨询会话



### 导航栏



## 基础数据结构（JS模型）



### 干系人

#### 基本属性：

ID、QQ、邮箱、电话、年龄、性别、地域、头像、名字

#### 类型：

角色类型：（VIP、企业）

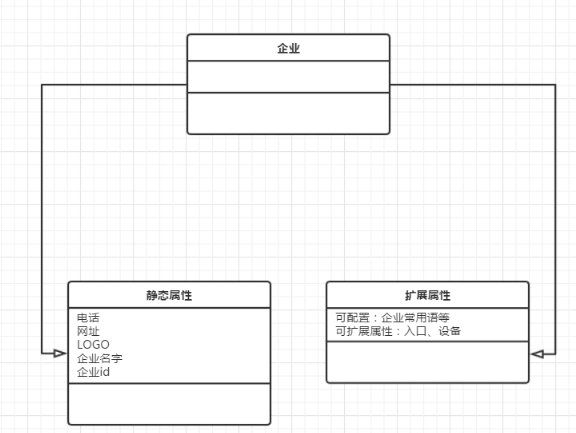
在线状态：在线、离线、忙碌、离开、隐身

加黑状态：加灰、审核中、通过；被谁加黑；加黑期限

等级

统计集合：会话数、在线时长

### 企业



#### 基本属性：

#### 企业id、企业名字、LOGO、网址、电话

#### 类型：

企业类型

### 业务咨询

#### 基本属性：

ID、干系人集、会话集、发消息、开始时间、结束时间、咨询状态、业务集

#### 类型：

咨询状态(销毁、挂起、进行中、初始化)

业务集(咨询总结、质检、评价、接待组、咨询发起页、商品详情页、是否vip、咨询过程是否、添加crm、咨询页面等级、是否购买咨询）

### 会话

#### 基本属性：

会话ID、会话token、创建时间、结束时间、会话有效期、会话生命周期状态、干系人集合、消息队列、最后会话人、最后离开人、会话类型

#### 类型：

会话有效期：(临时、永久)

会话生命周期状态：

1. 销毁：存库、销毁
2. 挂起：用户全部离开，待销毁或唤醒
3. 进行中：用户进入
4. 初始：会话创建，用户未进入

干系人集合：

1. 会话旁听人
2. 会话参与人
3. 会话发起人
4. 管理人（MASTER）

会话类型：

1. 转接
2. 邀请

### 消息

#### 基本属性：

消息ID、发送时间、送达超时时间、接收时间、发送人、接收人、阅后即焚标记、消息合法性、会话sessionid、token、消息类型、消息状态、消息发送通道

#### 消息类型：

1. 自定义扩展消息
2. 位置消息
3. 视频消息
4. 文件消息
5. 语音消息
6. 文本消息
7. 图片消息
8. 系统消息
9. 命令消息（撤回、已读、已送达、内容预知）

#### 消息状态：

1. 未达
2. 已焚
3. 已读
4. 已达

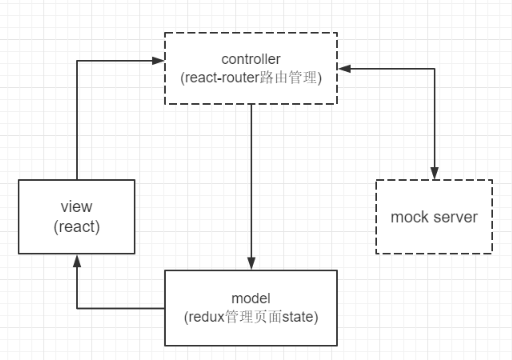
#### 消息发送通道：

1. 短信通道
2. 邮件通道
3. 离线通道
4. 在线通道

## 核心技术栈

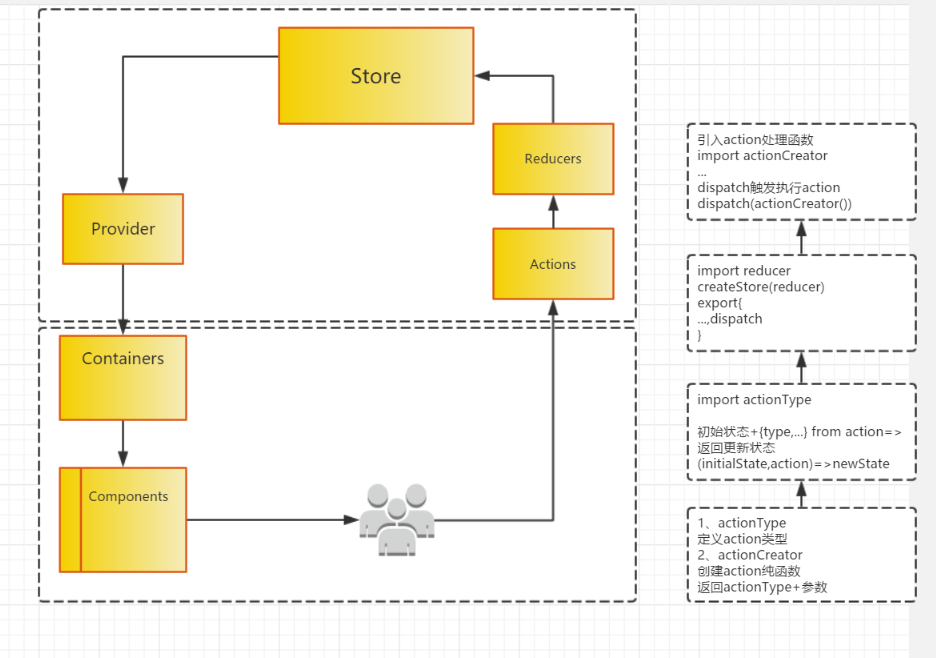
|  |  |
| --- | --- |
| 技术 | 用途 |
| es6（es2015） | 用 Javascript  ES6 标准来编写代码 |
| sass | CSS的一种语法，有很多新特性。更容易维护 |
| antd | 蚂蚁金服web UI组件 |
| antm | 蚂蚁金服移动端UI组件 |
| React | 用 React 组件写页面 |
| Node.js | 为 Electron 提供运行环境，也可以结合react server做后端直出 |
| redux | 数据流管理的框架 |
| Electron | 包装 HTML 页面，为网页提供一个客户端运行环境、支持多种系统环境 |
| EMQTT | 轻量级基于代理的布布/订阅的消息传输协仪 |
| react-router | 路由模块管理 |
| react-redux | 连接React和Redux |
| immutable.js | js 持久化数据框架，不可变对象 |
| isomorphic-fetch | Ajax服务调用 |
| Babel | 将ES6编译为ES5 |
| gulp+webpack | 打包，做自动化任务 |
| Mocha Chai、karma | 测试框架、自动化测试 |
| mock.js | 用于模拟后台数据的框架 |
| ESLint | JS语法检查 |
| Mixpanel | 跟踪用户操作 |
| jwplayer | 媒体视频播放 |
| mongoDB，mysql | 数据库 |
| Aotu.io（参考） | 项目流程工具：<https://athena.aotu.io/> |

## 前后端分离\*MVC

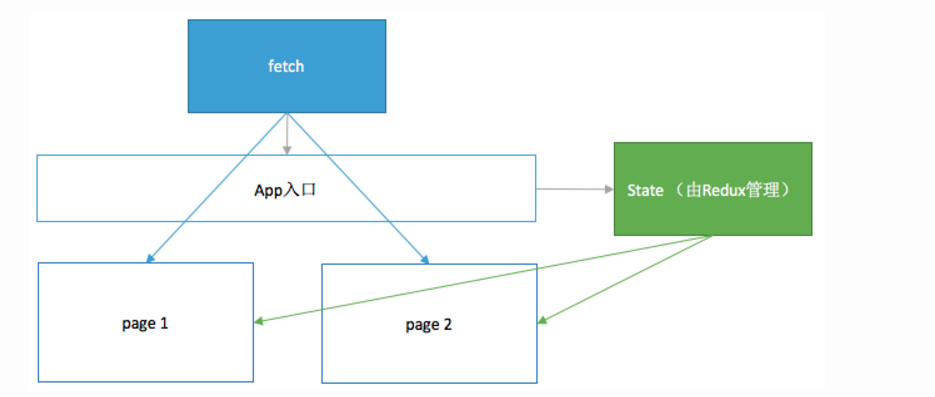


* view:使用react作为VIEW层的框架来渲染页面。可以借助node.js在服务端输出html,解决页面加载速度。(前后端同构：同一份component代码可以同时在前端和服务端同时运行)。
* Controller:使用react-router,它spa单页应用必备的利器、进行路由管理，处理请求页面地址的不同分发，应用到项目中也非常简单，它也有两种模式hash和pushstate切换页面。
* Model：redux来实现、配合不同的开源插件来实现强大的数据流管理。

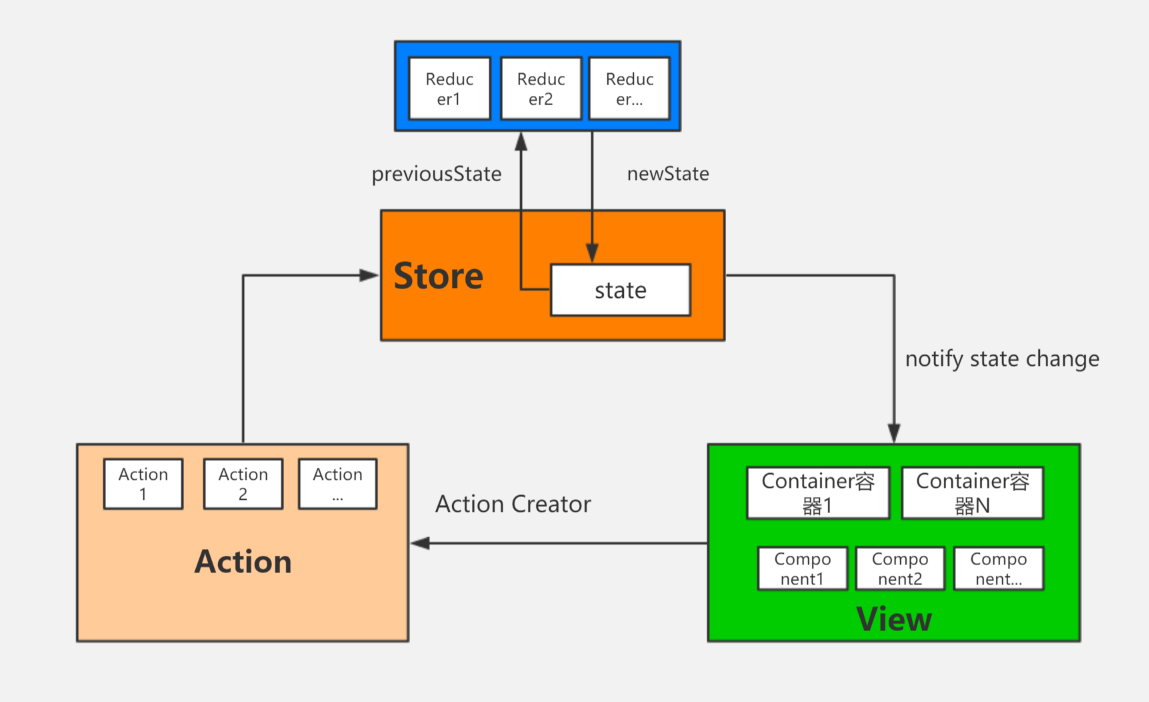
## 页面运行流程图



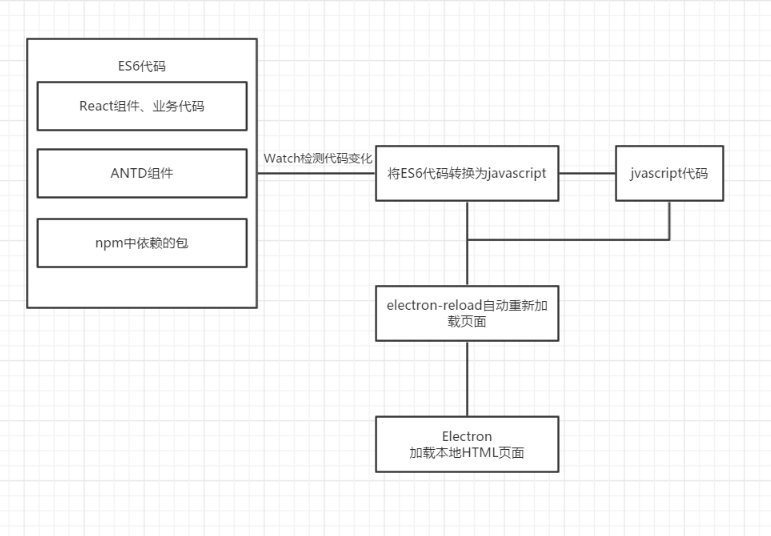
## 数据管理（redux）



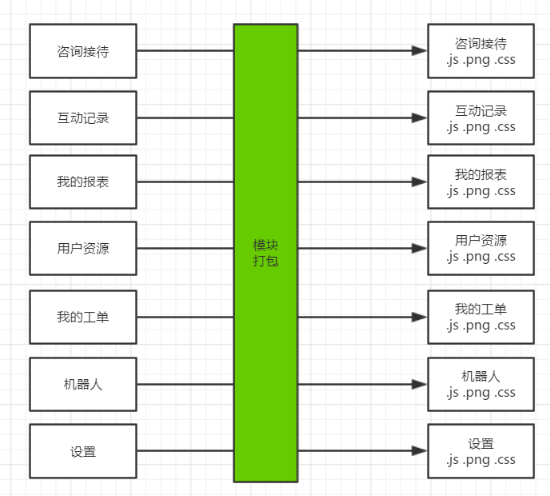
## 开发套路（Action->Reducer->Store）



## 编译流程图



## 模块打包(业务拆分)



## 文件结构

|  |  |
| --- | --- |
|  | containers 容器文件夹 ：存放容器组件  components 组件文件夹 ：存放普通显示组件   actions文件夹 ：存放可以发出的action  reducers 文件夹 ：存放action的处理器reducers  services 服务文件夹 ：存放经过封装的服务，如 ajax服务 , fetch服务，模拟数据服务  styles 样式文件夹 ：存放应用的样式，如css, scss  images 图片文件夹 : 存放图片资源  apis 开发接口文件夹 ： 存放开发接口文档 |

## Web服务请求(isomorphic-fetch)

### **fetch用法**

|  |  |  |
| --- | --- | --- |
| 方法 | 参数 | 说明 |
| Fetch | 参数1：URL、必填  参数2：init可选，它可以自定义HTTP请求中的一些部分。 | 论请求成功与否，它都返回一个 promise 对象，resolve 对应请求的 [Response](https://developer.mozilla.org/zh-CN/docs/Web/API/Response" \o "此页面仍未被本地化, 期待您的翻译!) |

### **Fetch 接口**

**[GlobalFetch](https://developer.mozilla.org/zh-CN/docs/Web/API/GlobalFetch" \o "The GlobalFetch interface of the Fetch API contains the GlobalFetch.fetch() method used to start the process of fetching a resource.)**

包含了 fetch() 方法，用于获取资源。

**[Headers](https://developer.mozilla.org/zh-CN/docs/Web/API/Headers" \o "此页面仍未被本地化, 期待您的翻译!)**

相当于 response/request 的头信息，可以修改它，或者针对不同的结果做不同的操作。

**[Request](https://developer.mozilla.org/zh-CN/docs/Web/API/Request" \o "此页面仍未被本地化, 期待您的翻译!)**

相当于一个资源请求。

**[Response](https://developer.mozilla.org/zh-CN/docs/Web/API/Response" \o "此页面仍未被本地化, 期待您的翻译!)**

相当于请求的响应

### **Fetch mixin**

**[Body](https://developer.mozilla.org/zh-CN/docs/Web/API/Body" \o "此页面仍未被本地化, 期待您的翻译!)**

提供了关联 response/request 中 body 的方法，可以定义它的文档类型以及请求如何被处理。

### **异步操作**

fetch的返回值是一个[Promise](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Promise" \t "http://reactnative.cn/docs/0.28/_blank)对象，你可以用两种办法来使用它：

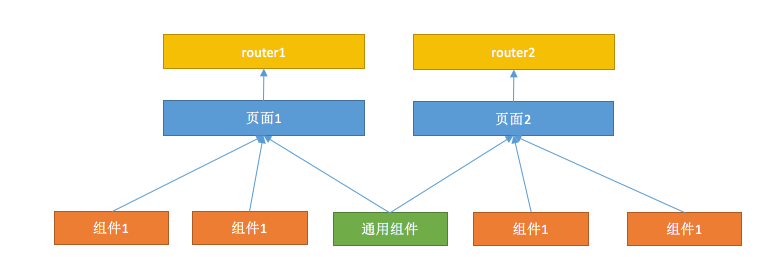
使用then和catch指定回调函数：

## **设计理念**

### **模块化**

当团队人员不断扩充时，我们需要制定统一的规范来对平时的开发工作做出一定约束和指导。统一的规范包括前端的代码规范，根据规范定义好一套代码检查的规则，在代码提交的时候进行检查，让开发人员知道自己的代码情况。

同时，根据以往的开发经验，我们制定了统一的项目框架，根据业务功能不同，将一个项目(app)拆分成不同的业务模块(module)，而每一个模块都包含自身的页面(page)以及构成页面所需要的组件(widget)，每一个项目涉及到app、module、page、widget这些已经约定好的概念，这样让项目结构更加清晰，而且让团队内不同业务的人员之间切换无障碍。



### **组件化**

在项目中引入组件化的概念，这里的组件对应上文讲到的widget，每一个组件都会包含组件自身的模板、css、js、图片以及说明文件，我们使用组件来拼装页面，像搭积木一样来拼装我们的页面，同时一个组件内可以调用另一个组件。

|  | **展示型组件** | **容器型组件** |
| --- | --- | --- |
| 组件设计目标 | 纯粹的UI(标签、样式) | 管理页面逻辑，和展示型组件 |
| 对Redux是否可见 | 不可见 | 可见 |
| 数据来源 | 从props中读取数据 | 通过Redux的state获取数据 |
| 改变数据的方式 | 调用通过props获取的回调函数 | 派发Redux的actions |
| 组件生成方式 | 手动写组件 | 通过React Redux动态生成 |

在拿到原型后，我们首先需要确定哪些需要做成公共组件，那些是要做成独立组件，以及组件间如何进行通信。在页面中调用这些组件后，会自动加载组件的模板以及组件的静态资源，而当组件不再需要时，只要移除掉组件引用，那么相应的模板和静态资源也会不再加载。

组件化的好处主要有这么几点

* 管理方便，我们可以把一个独立功能相关的文件在工程目录中放在一起，这样代码管理起来会非常便利
* 组件复用，通过抽取公共组件，可以实现组件复用，从而减少工作量，创造价值
* 分而治之，这是组件化最重要的一点，将页面组件化，就是对页面功能的拆分，将一个大的工程拆成小的零件，我们只需要关注每一个零件的功能，极大地降低了页面的开发与维护的难度

### **自动化编译**

在前端开发中，我们总是会去使用很多工具、手段来优化代码、提升开发效率，例如，我们会使用sass、less等CSS预处理工具来编写更好维护的样式代码，我们也会使用CSSLint、eslint等代码检查工具来检查代码的语法错误，使用文件合并压缩等手段来减少资源大小，除此之外我们还会去做雪碧图合并、多倍图处理、字体压缩处理、代码发布等等。

曾经有大神说过，超过90s的工作都应该自动化掉。而以上所有的这些工作，贯穿我们整个开发流程，但是不同工具的切换不但显得凌乱，而且影响开发效率。在自动化、工程编译的思想早已深入人心的当下，我们当然也要紧跟潮流，所以我们考虑通过自动化手段来提升我们的效率，让所有操作可以一键式开速执行完。

我们将通过定义好一系列的编译任务，按照一定顺序依次对我们的项目自动进行编译操作，最后产生出可上线的代码。

## **提升性能**

### 单页应用

客户端模块页，使用的就是单页架构。在这种架构下，基本不存在页面跳转的等待时间，只需要执行js逻辑触发界面变化，最多进行一次网络请求，获得服务端数据，其他资源均不需要再次请求。

### 资源离线

再快的网络交互，毕竟也是跨越了数个网络节点，因此一张图片、一个js，优化到了极致，也照样可能需要几百毫秒的时间来获得。因此想要打破这个极限，就要使用资源离线的策略。使用了这样的一个“离线包”策略。一些固定的图片、js库等，被打包放入app中（或根据需要，在app启动的时候进行下载更新）。

网页代码里面加载网络资源的需求，就变成了直接加载本地文件，速度自然得到再一次巨大的提升。

### 滚屏加载

滚屏加载是一种常见的无刷新动态加载数据的方案，通常用在列表形式数据展示中。一方面，数据不是通过翻页进行加载，这样就避免了再一次请求和渲染整个页面；另一方面，数据显示的数量是受限的，例如第一次只请求了10条数据，也就只需要渲染这10条数据，下拉滚屏的时候，再去获得下面10条数据。

### 静态资源预加载

### **前端流程工具**

#### 开发脚手架

前端流程工具 <https://athena.aotu.io/>

#### 功能一览

#### 创建项目

* 生成项目、模块、页面、组件文件结构

#### 编译预览

* 轻量组件化功能
* 根据组件加载情况生成资源依赖表
* 页面、组件html编译
* Sass/less 编译
* csslint/jshint 代码检查
* CSS合并压缩
* CSS prefix，px转rem
* JS合并压缩
* 自动生成雪碧图，自动多倍图
* 文件内联，自定义图片转base64
* 图片压缩
* 字体压缩
* 文件MD5戳
* 本地预览

# 模块详细设计：

## 登录:

现有登录流程：  


### **功能列表：**

1. 登录组件react
2. 登录验证（用户名、密码、siteID）
3. 安全&加密
4. 响应式
5. 非法用户处理
6. 获取服务端Token
7. 客服在线状态
8. 客户端版本信息
9. 最大登录次数
10. N次不通过（验证码登录）
11. 升级检查
12. 连接服务器验证

实现效果图:



 Action对象整理：用来代表所有会引起状态（state）变化的行为

|  |  |
| --- | --- |
| Action | 行为描述 |
| SIGNUP\_USER |  |
| SIGNIN |  |
| AUTH\_SIGNIN |  |
| AUTH\_SIGNIN\_SUCCESS |  |
| AUTH\_SIGNIN\_FAIL |  |
| AUTH\_SIGNOUT |  |
| AUTH\_SIGNOUT\_SUCCESS |  |
| AUTH\_SIGNOUT\_FAIL |  |
| AUTH\_SIGNUP\_SUCCESS |  |
| AUTH\_SIGNUP |  |
| AUTH\_SIGNUP\_FAIL |  |
| ...... |  |

api接口

|  |  |  |  |
| --- | --- | --- | --- |
| 方法名 | 参数 | 返回值 | 描述 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

### 实现方案：

app.js

router.js

login.js

auth.js

使用sessionStorage(包括Cookie，下略)或者localStroge保存从服务器获取的token，然后使用react-router中 onEnter 、onLogin方法依据sessionStorage或者localStroge中是否存在相应的token来判定登录状态(react-router的onEnter、onLogin调用requireAuth方法来判断auth.loggedIn()是否能正确返回localStorage.token,来维持登录状态)。

app.js:登录主文件，页面样式、路由处理、已登录用户跳转、属性、状态值传递等等

auth.js:使用了一个高级函数(high-order function，用例中为requireAuthentication)来包装需要登录权限的Compenent(用例中为ProtectedView)，这个Compenent位于所有需要登录权限的顶层：

参考资料：

[react-redux-jwt-auth-example](https://github.com/joshgeller/react-redux-jwt-auth-example" \t "http://www.tuicool.com/articles/_blank)

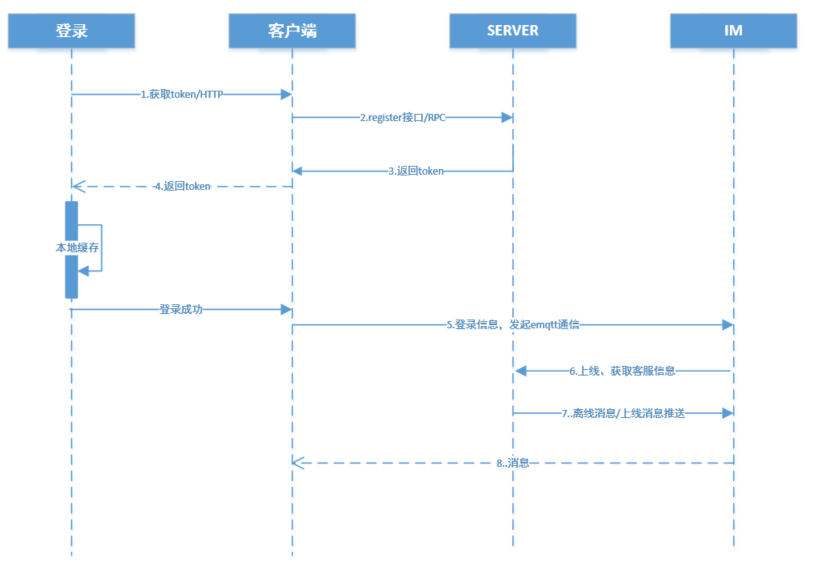
[redux-auth-wrapper](https://github.com/mjrussell/redux-auth-wrapper" \t "http://www.tuicool.com/articles/_blank)

## 首页：

### 功能列表：

1. 根据用户权限加载模块
2. 访客列表、我的同事列表、最近通话列表
3. 工作时长、登录时长
4. 消息提醒
5. 加载时设置用户的在线状态
6. 客服所在企业信息、权限信息获取
7. 上线、隐身、离线统计
8. Electron客户端
9. 网络环境监测
10. 客户端版本信息、运行环境
11. 退出

首页加载流程:



### 实现方案：

 Action对象整理：用来代表所有会引起状态（state）变化的行为

|  |  |
| --- | --- |
| Action | 行为描述 |
| ... |  |
|  |  |
|  |  |
|  |  |

根据页面开发容器组件、子组件。

State对象整理：

实现思路：设计state和action、及相关的reducer

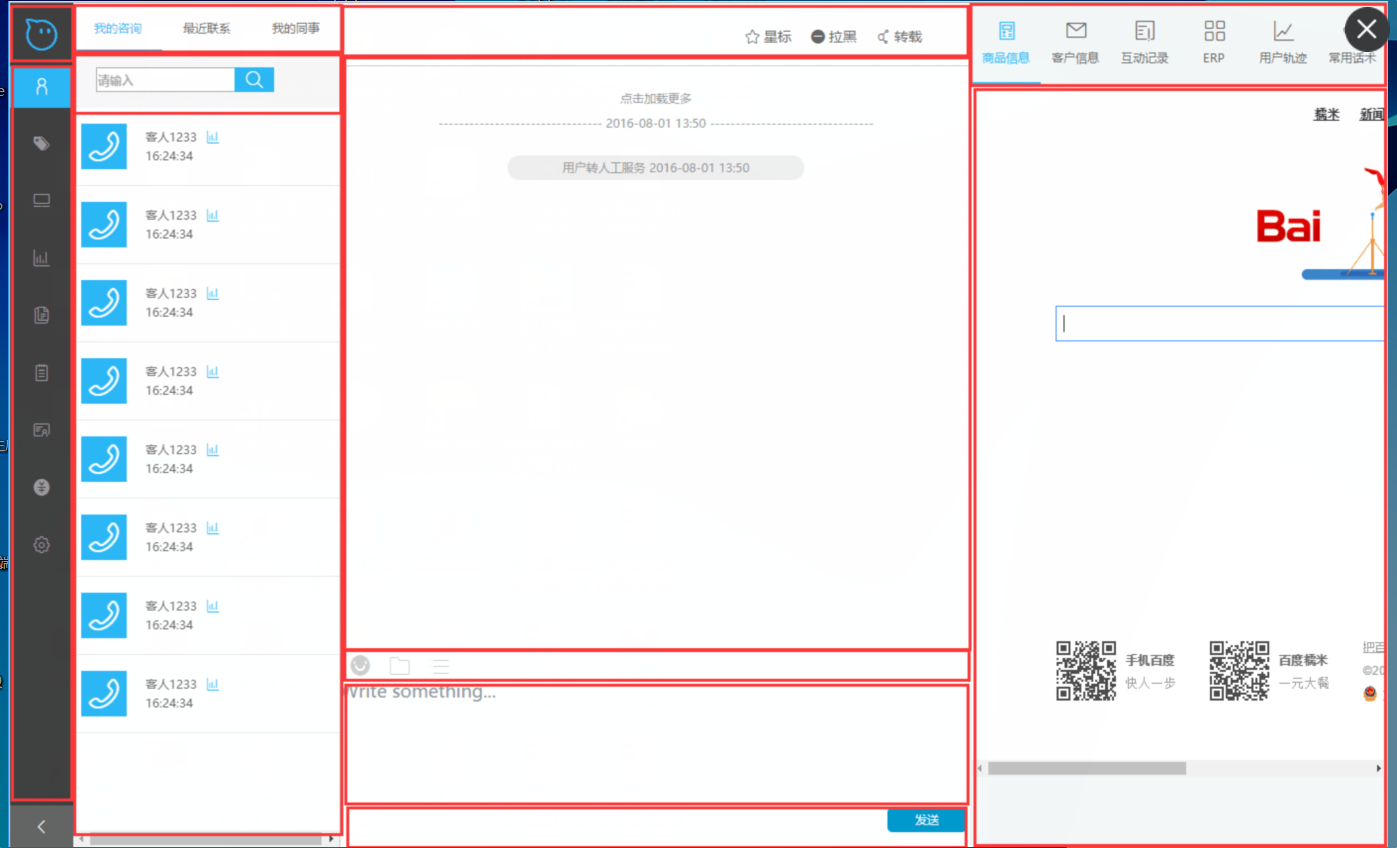
注意：每个组件只管理影响自己界面的state,在设计state和action之前，需要区分什么状态是组件独有的，什么状态是整个应有的，需要跨组件使用的，这些状态才使用redux管理。

## 待客云：

### **功能列表：**

1. 访客列表、我的同事列表、最近通话列表
2. 结束会话
3. 聊天窗口
4. 系统消息、机器人接入
5. 消息同步策略
6. 聊天工具栏（字体、表情、上传等）
7. 发起评价
8. 咨询总结
9. 生成工单
10. 转接机器人、客服（邀请）
11. 文件发送
12. 自定义表情（展现）
13. 截屏（没有更好方案，仿聊天咨询发起页）
14. 阻止访客
15. 人工辅助
16. 接管会话
17. 消息提醒
18. 咨询时长、在线时长
19. 消息同步策略
20. 轨迹......

实现效果：



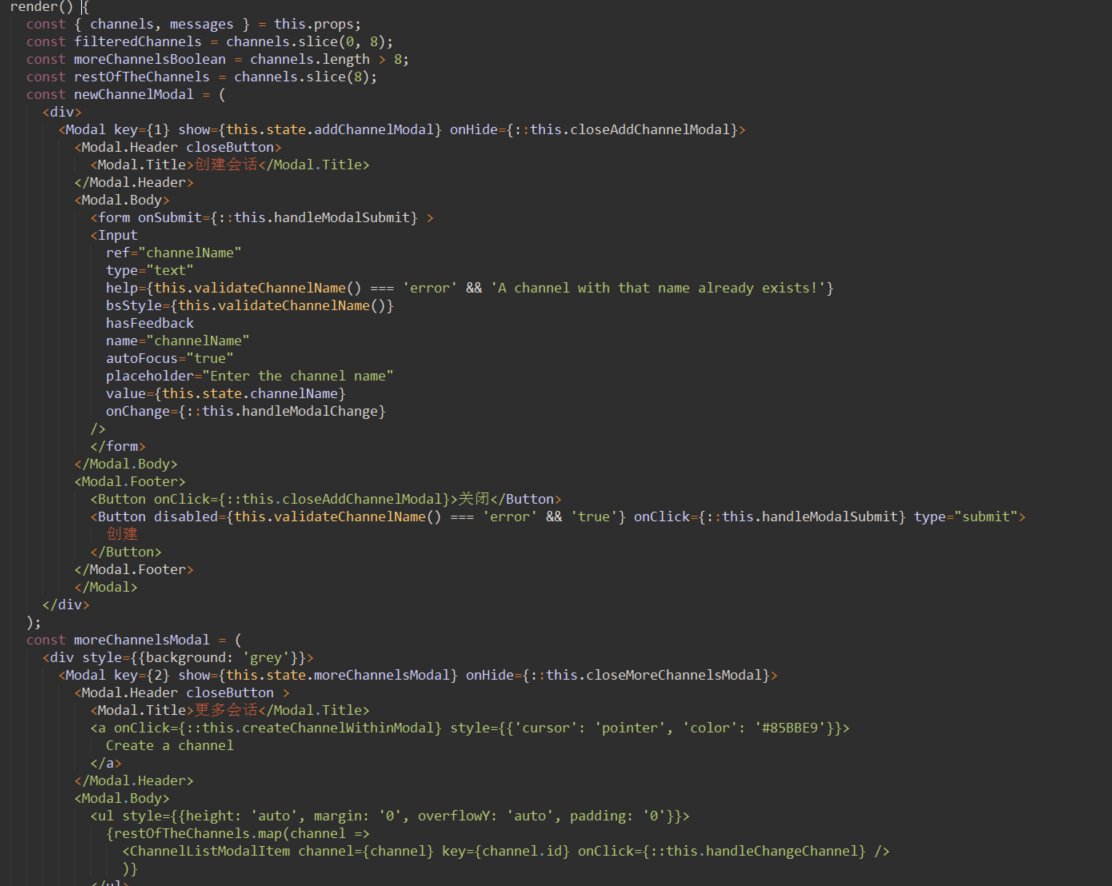
### 页面组件开发(container,component)

静态布局 - 使用 HTML + CSS 静态布局

* 布局：负责 contianer容器组件、component显示组件。普通显示组件放在容器组件内，由容器决定显示的值。
* 容组组件的格式（react组件）
* 做页面，SCSS

动态布局 - 使用 JSX 语法对静态布局做动态渲染处理

* 容器的state是store中state的一部分，根据定义好的store，将数据渲染页面上的静态内容。



* 普通显示组件的动态渲染

普通显示组件的动态渲染和容器类似，只是这里的state可以自己规定，使用组件的人按照示例传入数据即可。

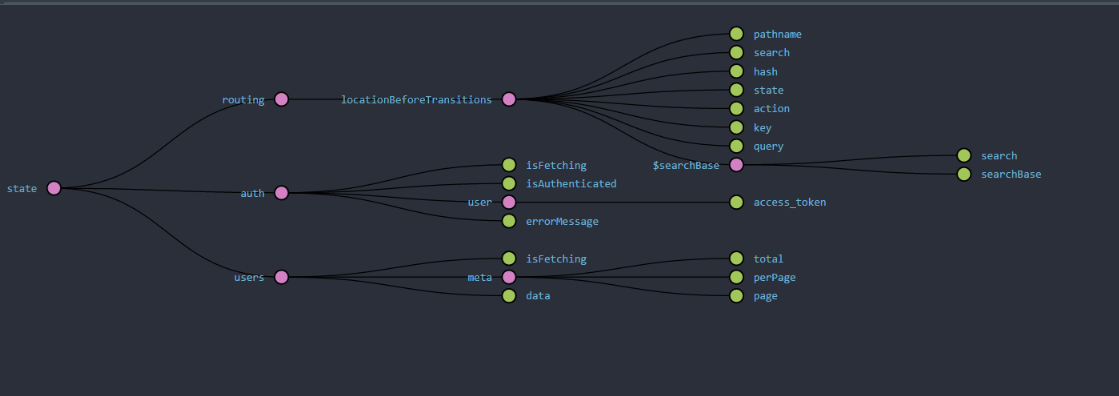
容器组件和普通显示组件state的对比：

1. 容器的state我们是从store中的总state直接获得的



select是一个state筛选器，功能是选择全集state中的一个state结点作为当前的容器state,而这个state的格式我们会在其对应的reducer中规定（reducer的参数中需传入state）

1. 普通显示组件的state格式由组件开发人员自己约定，也可以在模拟state给出默认数据，当别人要使用你的显示组件时，必须根据你规定的格式传入state数据。在组件里面，我们一般会实现如下这样一个自动切换器，当组件的使用者在使用该组件时没有传入state,就会显示内部模拟state.

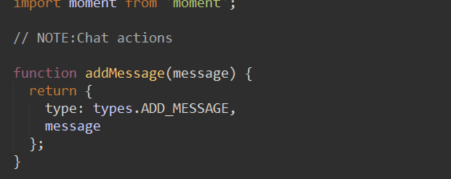


### 业务逻辑开发（reducer,action）

* 主要负责 action、reducer 、store部分

1. action 开发 - 制作 redux 流程的 action

action是一个简单对象、只要在对象内包含type属性指明action的名称。官方规定action的type要大写字母+下划线的格式、一个action只是一个对象，他要根据时机被store的dispatch函数调用才会开始进行处理。store.dispath(addMessage)



Action相关的技术栈、封装的中间件也比较多，但是思想是统一更好、更简便的管理。

1. reducer 开发 - 制作 redux 流程的 reducer

reducer是action的处理器，其开发的内容很清晰，就是业务函数，接受action和当前的state，返回新的state

技术要求：要求清楚redux数据流管理。

|  |  |
| --- | --- |
|  |  |

newState != oldState时才会使store触发react进行重新渲染

Reducer约束：在reducer里，不可以修改原来的state,需要保持使每个版本的state不变。

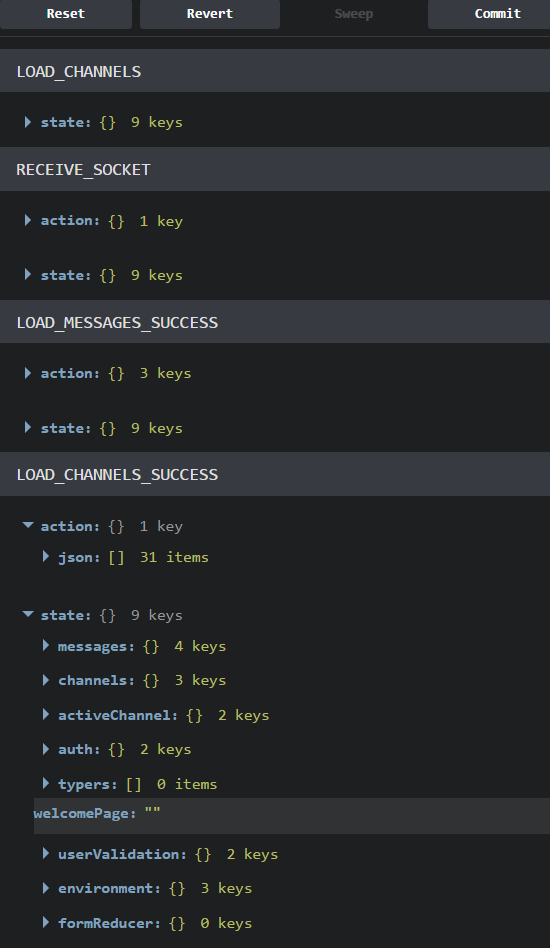
同理reducer相关的技术栈也比较多。。。

### 待客云实现方案：

 Action对象整理：用来代表所有会引起状态（state）变化的行为

|  |  |
| --- | --- |
| Action | 行为描述 |
| ADD\_MESSAGE | 添加消息 |
| RECEIVE\_MESSAGE | 接收消息 |
| ADD\_CHANNEL | 访客进入 |
| CHANGE\_CHANNEL | 切换访客 |
| LOAD\_MESSAGES | 加载会话 |
| LOAD\_MESSAGES\_SUCCESS | 加载会话完成 |
| LOAD\_MESSAGES\_FAIL | 加载会话失败 |
| LOAD\_CHANNELS | 加载访客 |
| LOAD\_CHANNELS\_SUCCESS | 加载访客成功 |
| LOAD\_CHANNELS\_FAIL | 加载访客失败 |
| LOAD\_USERSONLINE | 加载在线访客 |
| LOAD\_USERSONLINE\_SUCCESS | 加载在线访客成功 |
| LOAD\_USERSONLINE\_FAIL | 加载在线访客失败 |
| ...... |  |

State结构图(state.待客云)



#### 发送消息

#### 接收消息

#### 访客实时状态刷新（访客上线时，自动刷新访客列表）

示例：

<http://vuejs.github.io/js-repaint-perfs/react/>

<http://web-demo.adaptivecluster.com/#>

在超大量数据的首屏渲染速度上，React 有一定的渲染性能。以下是渲染的是 100 \* 5 的数据表，体验比较流畅。

对应的组件路径：src/component/TableList.js

#### 切换访客聊天列表

#### 在线时间

#### 加载访客信息（CRM）

#### 访问评价

功能列表：

|  |  |  |
| --- | --- | --- |
| 序号 | 功能标题 | 描述 |
|  |  |  |
|  |  |  |
|  |  |  |

实现方案：

#### 访客来源信息

#### 聊窗辅助工具

## 轨迹云：

### 功能列表：

1. 获取当前用户的轨迹信息，列表支持大数据展现
2. Api获取当前访客互动记录信息
3. Api获取当前访客CRM信息
4. 支持分页、排序
5. 互动记录支持语音、图片、链接、视频等富文本显示。
6. 查询

### 实现方案：

## 云平台：

### 功能列表：

### 实现方案：

## 权限：

### 功能列表：

### 实现方案：

## 设置：

### 功能列表：

### 实现方案：

## 前端组件(antd-ui)：

### **组件列表清单：**

* 登录组件
* 聊天窗口
* 富文本编辑器（simditor react）
* 列表组件
* 树
* 日期时间
* 按钮
* 单选
* 重选框
* 下拉多选&单选
* 模态窗口
* 页签
* 表格
* 分页
* 提示层
* 面板
* 通知提醒
* 菜单
* 工具条
* 查找组件
* 数值框
* <https://react.rocks/tag/List>

### 组件设计原则

### 如何开发一个表格组件



组件功能：

1. 按钮组
2. 搜索条件
3. 排序
4. 筛选
5. 状态点
6. 单行操作
7. 分页器／无限加载（可选）
8. 筛选

## Node.js：

node.js适用场景

1. 实时性应用，比如在线多人协作工具，网页聊天应用等。
2. 以I/O为主的高并发应用，比如为客户端提供API,读取数据库。
3. 流式应用，比如客户端经常上传文件。
4. 前后端分离。

注意点：

在利用Node.js编程时，任何耗时操作一定要使用异步来完成，避免阻塞当前函数。因为你为客户端提供服务，而所有代码总是单线程、顺序执行。

## Electron：

运行环境：

|  |  |
| --- | --- |
| Electron | 1.3.1 |
| Nodejs | 6.3.0 |
| Chrome | 52 |
| V8 | 5.2.361 |

功能列表：

1. 系统拖盘
2. 个性化界面
3. 消息提醒
4. ...

开源案例：

Electronic WeChat

<https://github.com/whphhg/vcash-electron>

介绍：

与 NW.js 相似，Electron 提供了一个能通过 JavaScript 和 HTML 创建桌面应用的平台，同时集成 Node 来授予网页访问底层系统的权限。

electron应用布署非常简单，存放应用程序的文件夹需要叫做 app 并且需要放在 Electron 的 资源文件夹下（在 macOS 中是指 Electron.app/Contents/Resources/，在 Linux 和 Windows 中是指resources/） 就像这样：

macOS:

electron/Electron.app/Contents/Resources/app/

├── package.json

├── main.js

└── index.html

在 Windows 和 Linux 中:

electron/resources/app

├── package.json

├── main.js

└── index.html

然后运行 Electron.app （或者 Linux 中的 electron，Windows 中的 electron.exe）, 接着 Electron 就会以你的应用程序的方式启动。

借助electron-quick-start开源项目快速开发基础模板。

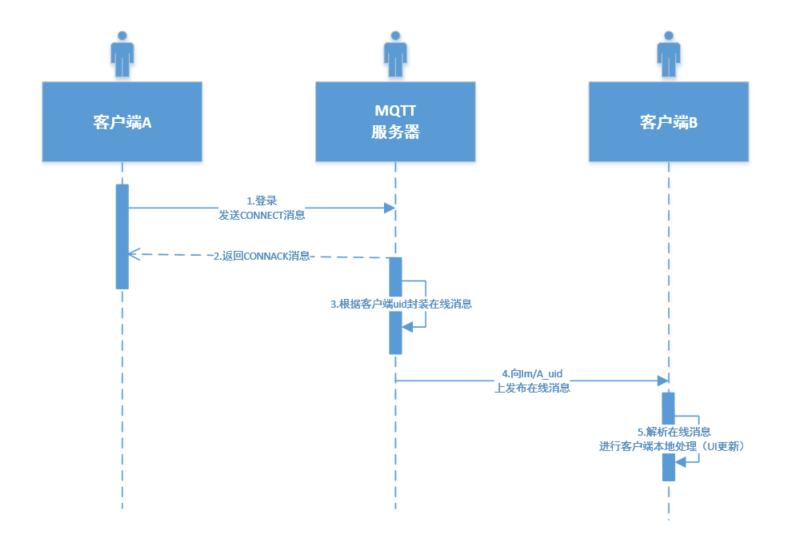
## MQTT:

MQTT协议作为一种基于发布/订阅模型的轻量级消息传输协义。可以分为三个部分：

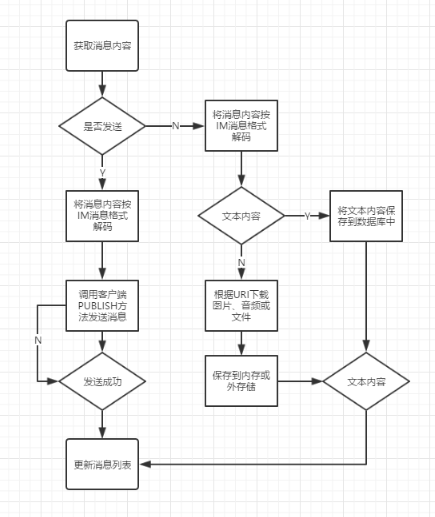
* **固定头部**：通用消息数据包格式
* **可变头部**：特定消息数据包格式
* **消息体**：有效载荷

### 话题预订阅序列图

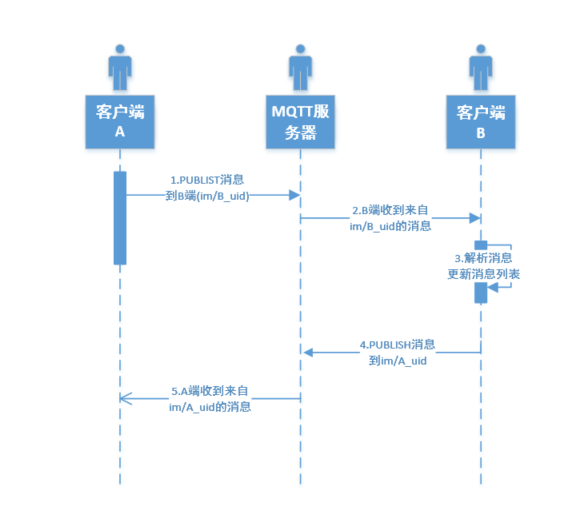
### 在线状态更新序列图



### 客户端对IM消息处理流程



### IM序列图



## 跨域:

## API接口设计：

## HTTPS:

## 升级：

## 国际化：

LocaleProvider为组件内建文案提供统一的国际化支持。

使用[#](http://ant.design/components/locale-provider/" \l "使用)

LocaleProvider 使用 React 的 [context](https://facebook.github.io/react/docs/context.html) 特性，只需在应用外围包裹一次即可全局生效。

import enUS from 'antd/lib/locale-provider/en\_US';

return <**LocaleProvider locale**={enUS}><App /></**LocaleProvider**>;

## 响应式：

## 运行监控：

## 自动化测试：

## Nginx：

## 工程化：

开发模式(dev)：

线上模式(prod)：

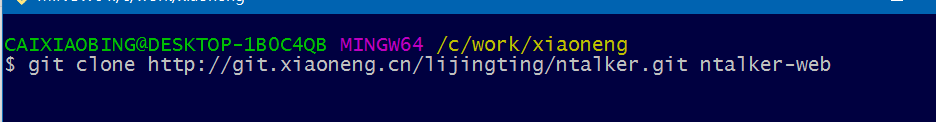
# 开发环境搭建：

## 环境准备

* node ^4.2.0
* npm ^3.0.0
* Git^2.8.4

## 下载代码环境

git地址：[git@git.xiaoneng.cn:lijingting/ntalker.git](mailto:git@git.xiaoneng.cn:lijingting/ntalker.git)



1. $ npm install 安装依赖插件
2. $ npm start启动web服务器
3. $npm start:mock 启动mock数据服务

npm描述：

| npm run <script> | 描述 |
| --- | --- |
| start:prod | 运行在线上环境 |
| start:mock | 启动MOCK服务 |
| lint | JS语法检查 |
| lint:watch | JS语法检查.实时检查 |
| build | 资源文件打包，调用WEBPACK |
| test | 运行测试用例 |
| ...... |  |

## 运行产品

运行浏览器，node下运行npm start即可打开产品[http://localhost:3000](http://localhost:3000rs)

## 开发工具：

IDE:web strom

Chrome插件:Redux DevTools、React Developer Tools

# WEB聊窗前端优化方案：

* 预期目标：个人认为前端部分受环境限制，很难于借助一些框架来重构（浏览器兼容、文件大小、网络传输等方面）。
* 样式复杂、导致难以维护。可以借助一些新的SCSS规范来编写。

具体技术方案：

1. 抽离DOM中的样式，换成变量写法（LESS,CSS,SASS）如下。

开发中使用的样式表

description {

margin-bottom: 20px;

font-size: 18px;

text-align: center;

color: #656656;

}

container {

padding: 30px;

margin-top: 65px;

align-items: center;

display: block;

}

使用react-css等一些插件来生成style.js文件。

//引用CSS样式文件

var styles = require('./style.js') //{"description":{"marginBottom":20,"fontSize":18,"textAlign":"center","color":"#656656"},

"container":{"padding":30,"marginTop":65,"alignItems":"center"}}

class ChatWindowPage extends Component {

render() {

return (

<View style={**styles.container**}>

<Text style=**{styles.description}**>

</Text>

</View>

);

}

}

* 后期的主要精力可以借助一些开发工具做自动化的生成、打包(一键发布工具)。

# 移动端重构：

移动端总结构图

## 页面布局：

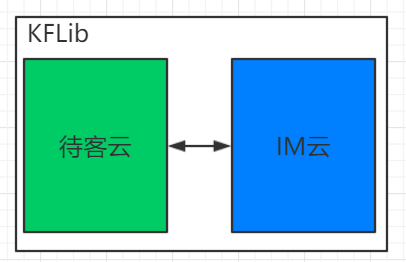
## 业务共用：action,store,reducer是客户端的子集。

**移动端UI组件：**

antm是蚂蚁金服提供的移动端ui组件，基于react native

# 业务API：

### KFLib.js(通讯能力库)

` 

#### Model（JS模型）:

命名空间：kflib.model

##### 会话类：Conversation

|  |  |  |  |
| --- | --- | --- | --- |
| **类名** |  | | |
| 方法 | 返回值 | 参数 | 描述 |
| getConversationTitle() | string |  | 获取会话标题 |
| getConversationTemplateId | Int |  | 获取会话模板 |
| getConversationTemplateType | string |  | 获取会话模板类型 |
| getConversationType() | string |  | 获取会话类型 |
| getLatestMessage() | enum |  | 获取本会话最后一条消息 |
| getLatestMessageId() | int |  | 获取本会话最后一条消息 Id |
| getNotificationStatus() | enum |  | 获取通知消息状态 |
| getObjectName() | string |  | 获取消息对象名称 |
| getReceivedStatus() | enum |  | 获取接收到的消息的状态 |
| getReceivedTime() | date |  | 获取消息接收时间 |
| getSenderUserId() | string |  | 获取发送消息的用户 Id |
| getSenderUserName() | string |  | 获取发送消息的用户名称 |
| getSentStatus() | enum |  | 获取发送出的消息的状态 |
| getSentTime() | date |  | 获取最后消息发送时间 |
| getTargetId() | string |  | 获取目标 Id |
| getUnreadMessageCount() | int |  | 获取未读消息数 |
| isTop() | boolean |  | 获取置顶状态 |
| setConversationTitle | void |  | 设置会话标题 |
| setConversationType | void |  | 设置会话类型 |
| setLatestMessage | void | MessageContent | 设置本会话最后一条消息 |
| setLatestMessageId | void | int | 设置本会话最后一条消息 Id |
| setNotificationStatus | void | ConversationNotificationStatus | 设置获取通知消息状态 |
| setObjectName | void | string | 设置消息对象名称 |
| setReceivedStatus | void | string | 设置接收到的消息的状态 |
| setReceivedTime | void | Longtime | 设置消息接收时间 |
| SetSenderUserId | void | string | 设置发送消息的用户 Id |
| setSenderUserName | void | string | 设置发送消息的用户名称 |
| setSentStatus | void | Message.SentStatus | 设置发送出的消息的状态 |
| setSentTime | void | Long time | 设置最后消息发送时间 |
| setTargetId | void | string | 设置目标 Id |
| setTop | void | bool | 设置置顶状态 |
| setUnreadMessageCount | void | int | 设置未读消息数 |

##### 消息类：Message

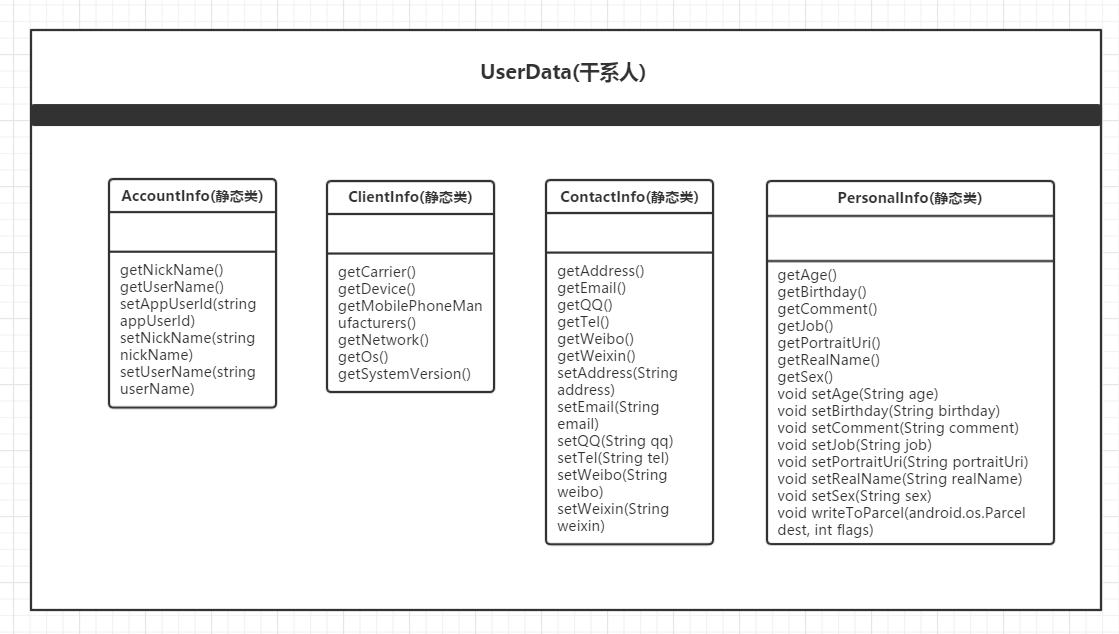
|  |  |  |  |
| --- | --- | --- | --- |
| **类名** |  | | |
| 方法 | 返回值 | 参数 | 描述 |
| getContent() | MessageContent |  | 获取消息内容 |
| getConversationType() | Conversation.ConversationType |  | 获取会话类型 |
| getConversationTypeTemplate() | Conversation.ConversationTypeTemplate |  | 获取会话类型模板 |
| getExtra() | String |  | 获取消息的附加信息 |
| getMessageDirection() | Message.MessageDirection |  | 获取消息方向 |
| getMessageId() | int |  | 获取消息 Id |
| getObjectName() | string |  | 获取消息对象名称 |
| getReceivedStatus() | Message.ReceivedStatus |  | 获取接收到的消息的状态 |
| getReceivedTime() | date |  | 获取消息接收时间 |
| getSenderUserId() | string |  | 获取发送消息的用户 Id |
| getSentStatus() | Message.SentStatus |  | 获取发送出的消息的状态 |
| getSentTime() | date |  | 获取消息发送时间 |
| getTargetId() | String |  | 获取目标 Id |
| getUId() | String |  | 获取 message uid |
| setContent() | void | Conversation.ConversationType,  MessageContent | 设置消息内容 |
| setConversationType() | void | Conversation.ConversationType | 设置会话类型 |
| setExtra() | void | string | 设置消息的附加信息 |
| setMessageDirection() | void | Message.MessageDirection | 设置消息方向 |
| setMessageId() | void | int | 设置消息 Id |
| setObjectName() | void | string | 设置消息对象名称 |
| setReceivedStatus() | void | Message.ReceivedStatus | 设置接收到的消息的状态 |
| setReceivedTime() | void | date | 设置消息接收时间 |
| setSenderUserId | void | string | 设置发送消息的用户 Id |
| setSentStatus | void | Message.SentStatus | 设置发送出的消息的状态 |
| setSentTime | void | date | 设置消息发送时间 |
| setTargetId | void | string | 设置目标 Id |
| setUId | void | string | 设置 message uid |

##### 消息接收状态类：Message.ReceivedStatus

|  |  |  |  |
| --- | --- | --- | --- |
| **类名** |  | | |
| 方法 | 返回值 | 参数 | 描述 |
| getFlag() | Int |  | 获取状态标识 |
| isDownload() | bool |  | 获取文件是否已经下载的状态 |
| isListened() | bool |  | 获取是否已被收听的状态 |
| isMultipleReceive() | bool |  | 获取是否被其他端同时接收 |
| isRead() | bool |  | 获取是否已读取的状态 |
| isRetrieved() | bool |  | 获取是否已经被收取过 |
| setDownload() | void |  | 设置文件是否已经下载的状态 |
| setListened() | void |  | 设置是否已被收听的状态 |
| setMultipleReceive() | void |  | 设置为已被其他端同时接收 |
| setRead() | void |  | 设置是否已读取的状态 |
| setRetrieved() | void |  | 设置为已经被收取 |

##### 干系人类

类名：UserData



##### 聊天室类：ChatRoomInfo

|  |  |  |  |
| --- | --- | --- | --- |
| **类名** |  | | |
| 方法 | 返回值 | 参数 | 描述 |
| getChatRoomId | string |  | 获取聊天室ID |
| setChatRoomId | void |  | 设置聊天室ID |
| getMemberOrder | Enum |  | 获取聊天室成员（排序） |
| setMemberOrder | void |  | 设置聊天室成员 |
| setTotalMemberCount | void |  | 设置聊天室成员总数 |
| getMemberInfo | array |  | 获取聊天室成员信息 |
| setMemberInfo | void |  | 设置聊天室成员信息 |
| getTotalMemberCount | int |  | 获取聊天室成员总数 |
|  |  |  |  |

##### 群组:Group

|  |  |  |  |
| --- | --- | --- | --- |
| **类名** |  | | |
| 方法 | 返回值 | 参数 | 描述 |
| getId() | string |  | 获取群组 Id |
| getName() | string |  | 获取群组名称 |
| getPortraitUri() | string |  | 获取群组头像 |
| setId() | void | string | 设置群组 Id |
| setName() | void | string | 设置群组名称 |
| setPortraitUri() | void | string | 设置群组头像 |
|  |  |  |  |

##### 讨论组:Discussion

|  |  |  |  |
| --- | --- | --- | --- |
| **类名** |  | | |
| 方法 | 返回值 | 参数 | 描述 |
| getCreatorId() | string |  | 获取讨论组创建者 Id |
| getId() | string |  | 获取讨论组 Id |
| getMemberIdList() | Array/json |  | 获取讨论组成员 Id 列表 |
| getName() | string |  | 获取讨论组名称 |
| isOpen() | bool |  | 获取讨论组是否公开好友邀请 |
| setCreatorId() |  | string | 设置讨论组创建者 Id |
| setId() |  | string | 设置讨论组 Id |
| setMemberIdList() |  | array | 设置讨论组成员 Id 列表 |
| setName() |  | string | 设置讨论组名称 |
| setOpen() |  | bool | 设置讨论组是否公开好友邀请 |

枚举：

* **[ChatRoomInfo.ChatRoomMemberOrder](http://www.rongcloud.cn/docs/api/android/imlib/io/rong/imlib/model/ChatRoomInfo.ChatRoomMemberOrder.html" \o "io.rong.imlib.model中的枚举" \t "http://www.rongcloud.cn/docs/api/android/imlib/io/rong/imlib/model/classFrame)（**排序方式**）**

|  |  |
| --- | --- |
| [RC\_CHAT\_ROOM\_MEMBER\_ASC](http://www.rongcloud.cn/docs/api/android/imlib/io/rong/imlib/model/ChatRoomInfo.ChatRoomMemberOrder.html" \l "RC_CHAT_ROOM_MEMBER_ASC) | 升序，返回最早加入的用户列表. |
| [RC\_CHAT\_ROOM\_MEMBER\_DESC](http://www.rongcloud.cn/docs/api/android/imlib/io/rong/imlib/model/ChatRoomInfo.ChatRoomMemberOrder.html" \l "RC_CHAT_ROOM_MEMBER_DESC) | 降序，返回最晚加入的用户列表. |

* **[Conversation.ConversationNotificationStatus](http://www.rongcloud.cn/docs/api/android/imlib/io/rong/imlib/model/Conversation.ConversationNotificationStatus.html" \o "io.rong.imlib.model中的枚举" \t "http://www.rongcloud.cn/docs/api/android/imlib/io/rong/imlib/model/classFrame)（**会话提醒状态**）**

|  |  |
| --- | --- |
| [DO\_NOT\_DISTURB](http://www.rongcloud.cn/docs/api/android/imlib/io/rong/imlib/model/Conversation.ConversationNotificationStatus.html" \l "DO_NOT_DISTURB) | 免打扰 |
| [NOTIFY](http://www.rongcloud.cn/docs/api/android/imlib/io/rong/imlib/model/Conversation.ConversationNotificationStatus.html" \l "NOTIFY) | 提醒 |

* **[Conversation.ConversationType](http://www.rongcloud.cn/docs/api/android/imlib/io/rong/imlib/model/Conversation.ConversationType.html" \o "io.rong.imlib.model中的枚举" \t "http://www.rongcloud.cn/docs/api/android/imlib/io/rong/imlib/model/classFrame)（**会话类型枚举**）**

|  |  |
| --- | --- |
| [CHATROOM](http://www.rongcloud.cn/docs/api/android/imlib/io/rong/imlib/model/Conversation.ConversationType.html" \l "CHATROOM) | 聊天室 |
| [CUSTOMER\_SERVICE](http://www.rongcloud.cn/docs/api/android/imlib/io/rong/imlib/model/Conversation.ConversationType.html" \l "CUSTOMER_SERVICE) | 客服 |
| [DISCUSSION](http://www.rongcloud.cn/docs/api/android/imlib/io/rong/imlib/model/Conversation.ConversationType.html" \l "DISCUSSION) | 讨论组 |
| [GROUP](http://www.rongcloud.cn/docs/api/android/imlib/io/rong/imlib/model/Conversation.ConversationType.html" \l "GROUP) | 群组 |
| [PRIVATE](http://www.rongcloud.cn/docs/api/android/imlib/io/rong/imlib/model/Conversation.ConversationType.html" \l "PRIVATE) | 私聊 |
| [PUSH\_SERVICE](http://www.rongcloud.cn/docs/api/android/imlib/io/rong/imlib/model/Conversation.ConversationType.html" \l "PUSH_SERVICE) | push推送 |
| [SYSTEM](http://www.rongcloud.cn/docs/api/android/imlib/io/rong/imlib/model/Conversation.ConversationType.html" \l "SYSTEM) | 系统 |

* **[CustomServiceMode](http://www.rongcloud.cn/docs/api/android/imlib/io/rong/imlib/model/CustomServiceMode.html" \o "io.rong.imlib.model中的枚举" \t "http://www.rongcloud.cn/docs/api/android/imlib/io/rong/imlib/model/classFrame)（客服服务模式）**

|  |  |
| --- | --- |
| [CUSTOM\_SERVICE\_MODE\_HUMAN](http://www.rongcloud.cn/docs/api/android/imlib/io/rong/imlib/model/CustomServiceMode.html" \l "CUSTOM_SERVICE_MODE_HUMAN) | 人工模式 |
| [CUSTOM\_SERVICE\_MODE\_HUMAN\_FIRST](http://www.rongcloud.cn/docs/api/android/imlib/io/rong/imlib/model/CustomServiceMode.html" \l "CUSTOM_SERVICE_MODE_HUMAN_FIRST) | 人工优先模式 |
| [CUSTOM\_SERVICE\_MODE\_NO\_SERVICE](http://www.rongcloud.cn/docs/api/android/imlib/io/rong/imlib/model/CustomServiceMode.html" \l "CUSTOM_SERVICE_MODE_NO_SERVICE) | 无服务 |
| [CUSTOM\_SERVICE\_MODE\_ROBOT](http://www.rongcloud.cn/docs/api/android/imlib/io/rong/imlib/model/CustomServiceMode.html" \l "CUSTOM_SERVICE_MODE_ROBOT) | 机器人模式 |
| [CUSTOM\_SERVICE\_MODE\_ROBOT\_FIRST](http://www.rongcloud.cn/docs/api/android/imlib/io/rong/imlib/model/CustomServiceMode.html" \l "CUSTOM_SERVICE_MODE_ROBOT_FIRST) | 机器人优先模式，在此模式下，可以切换至人工模式 |

* **[Message.MessageDirection](http://www.rongcloud.cn/docs/api/android/imlib/io/rong/imlib/model/Message.MessageDirection.html" \o "io.rong.imlib.model中的枚举" \t "http://www.rongcloud.cn/docs/api/android/imlib/io/rong/imlib/model/classFrame)（消息方向）**

|  |  |
| --- | --- |
| [RECEIVE](http://www.rongcloud.cn/docs/api/android/imlib/io/rong/imlib/model/Message.MessageDirection.html" \l "RECEIVE) | 接收消息 |
| [SEND](http://www.rongcloud.cn/docs/api/android/imlib/io/rong/imlib/model/Message.MessageDirection.html" \l "SEND) | 发送消息 |

* **[Message.SentStatus](http://www.rongcloud.cn/docs/api/android/imlib/io/rong/imlib/model/Message.SentStatus.html" \o "io.rong.imlib.model中的枚举" \t "http://www.rongcloud.cn/docs/api/android/imlib/io/rong/imlib/model/classFrame)（发送状态）**

|  |  |
| --- | --- |
| [DESTROYED](http://www.rongcloud.cn/docs/api/android/imlib/io/rong/imlib/model/Message.SentStatus.html" \l "DESTROYED) | 对方已销毁 |
| [FAILED](http://www.rongcloud.cn/docs/api/android/imlib/io/rong/imlib/model/Message.SentStatus.html" \l "FAILED) | 发送失败 |
| [READ](http://www.rongcloud.cn/docs/api/android/imlib/io/rong/imlib/model/Message.SentStatus.html" \l "READ) | 对方已读 |
| [RECEIVED](http://www.rongcloud.cn/docs/api/android/imlib/io/rong/imlib/model/Message.SentStatus.html" \l "RECEIVED) | 对方已接收 |
| [SENDING](http://www.rongcloud.cn/docs/api/android/imlib/io/rong/imlib/model/Message.SentStatus.html" \l "SENDING) | 发送中 |
| [SENT](http://www.rongcloud.cn/docs/api/android/imlib/io/rong/imlib/model/Message.SentStatus.html" \l "SENT) | 已发送 |

#### KFLib核心库

1. **初始化小能云通讯SDK**

**Init**

**事例：**

**//new KFLib().init**

1. **初始化小能云SDK**

**InitWithAppKey**

**事例：**

1. **设置IM云服务器、上传文件服务器信息**

**setServerInfo()**

1. **与小能云服务器建立连接**

**connectWidthToken(params)**

1. **断开与小能云服务器的连接**

**disconnect**

1. **断开与小能云服务器的连接，但仍然接收远程推送**

**disconnect(params)**

1. **断开与小能云服务器的连接，并不再接收远程推送**

**Logout**

1. **设置KFLib的连接状态监听器**

**setKFConnectionStatusChangeDelegate**

1. **获取当前SDK的连接状态**

**getConnectionStatus(params)**

1. **获取当前的网络状态**

**getCurrentNetworkStatus**

1. **SDK当前所处的运行状态**

**sdkRunningMode(params)**

1. **当前登录用户的用户信息**

**currentUserInfo(params)**

1. **从小能云服务器获取用户信息**

**getUserInfo(params)**

1. **注册自定义的消息类型**

**registerMessageType(params)**

1. **发送消息**

**sendMessage(params,ref)**

1. **发送图片消息**

**sendImageMessage(params,targetId,content.....)**

1. **发送图片消息（到指定服务器）**

**sendImageMessage(params)**

1. **插入消息**

**insertMessage(params,targetId)**

1. **下载消息内容中的媒体信息**

**downloadMediaFile(params,...)**

1. **发送状态消息**

**sendStatusMessage(targetId,content,status)**

1. **发送消息**

**sendMessage(targetIddd,content)**

1. **发送图片消息**

**sendImageMessage(targetId,content,pushContent)**

1. **设置KFLib的消息接收监听器**

**setReceiveMessageDelegate(object)**

1. **发送某个会话中消息阅读的回执**

**sendReadReceiptMessage(targetId,timer)**

1. **撤回消息**

**recallMessage(params)**

1. **获取某个会话中指定数量的最新消息实体**

**getLastMessages(targetId)**

1. **获取会话中，从指定消息之前、指定数量的最新消息实体**

**getHistoryMessages(targetId)**

1. **获取会话中，从指定消息之前、指定数量的、指定消息类型的最新消息实体**

**getHistoryMessages(targetId,object)**

1. **获取会话中，从指定消息之前、指定数量的、指定消息类型、可以向前或向后查找的最新消息实体**

**[getHistoryMessages(targetId:objectName:baseMessageId:isForward:count:](http://rongcloud.cn/docs/api/ios/imlib/Classes/RCIMClient.html" \l "//api/name/getHistoryMessages:targetId:objectName:baseMessageId:isForward:count:)**

1. **从服务器端获取之前的历史消息**

**[getRemoteHistoryMessages(targetId:recordTime:count](http://rongcloud.cn/docs/api/ios/imlib/Classes/RCIMClient.html" \l "//api/name/getRemoteHistoryMessages:targetId:recordTime:count:success:error:)**

1. **获取会话中@提醒自己的消息**

**[getUnreadMentionedMessages(targetId:](http://rongcloud.cn/docs/api/ios/imlib/Classes/RCIMClient.html" \l "//api/name/getUnreadMentionedMessages:targetId:)**

1. **获取消息的发送时间（Unix时间戳、毫秒）**

**[getMessageSendTime(](http://rongcloud.cn/docs/api/ios/imlib/Classes/RCIMClient.html" \l "//api/name/getMessageSendTime:))**

1. **通过messageId获取消息实体**

**[getMessage(](http://rongcloud.cn/docs/api/ios/imlib/Classes/RCIMClient.html" \l "//api/name/getMessage:))**

1. **通过全局唯一ID获取消息实体**

**[getMessageByUId(](http://rongcloud.cn/docs/api/ios/imlib/Classes/RCIMClient.html" \l "//api/name/getMessageByUId:))**

1. **删除消息**

**[deleteMessages(](http://rongcloud.cn/docs/api/ios/imlib/Classes/RCIMClient.html" \l "//api/name/deleteMessages:))**

1. **删除某个会话中的所有消息**

**[deleteMessages(targetId)](http://rongcloud.cn/docs/api/ios/imlib/Classes/RCIMClient.html" \l "//api/name/deleteMessages:targetId:success:error:)**

1. **删除某个会话中的所有消息**

**[clearMessages(targetId)](http://rongcloud.cn/docs/api/ios/imlib/Classes/RCIMClient.html" \l "//api/name/clearMessages:targetId:)**

1. **设置消息的附加信息**

**[setMessageExtra(value)](http://rongcloud.cn/docs/api/ios/imlib/Classes/RCIMClient.html" \l "//api/name/setMessageExtra:value:)**

1. **设置消息的接收状态**

**[setMessageReceivedStatus(receivedStatus)](http://rongcloud.cn/docs/api/ios/imlib/Classes/RCIMClient.html" \l "//api/name/setMessageReceivedStatus:receivedStatus:)**

1. **设置消息的发送状态**

**[setMessageSentStatus(sentStatus)](http://rongcloud.cn/docs/api/ios/imlib/Classes/RCIMClient.html" \l "//api/name/setMessageSentStatus:sentStatus:)**

1. **获取会话列表**

**[getConversationList(](http://rongcloud.cn/docs/api/ios/imlib/Classes/RCIMClient.html" \l "//api/name/getConversationList:))**

1. **获取单个会话数据**

**[getConversation(targetId)](http://rongcloud.cn/docs/api/ios/imlib/Classes/RCIMClient.html" \l "//api/name/getConversation:targetId:)**

1. **获取会话中的消息数量**

**[getMessageCount(targetId)](http://rongcloud.cn/docs/api/ios/imlib/Classes/RCIMClient.html" \l "//api/name/getMessageCount:targetId:)**

1. **删除指定类型的会话**

**[clearConversations(](http://rongcloud.cn/docs/api/ios/imlib/Classes/RCIMClient.html" \l "//api/name/clearConversations:))**

1. **从本地存储中删除会话**

**[removeConversation（targetId）](http://rongcloud.cn/docs/api/ios/imlib/Classes/RCIMClient.html" \l "//api/name/removeConversation:targetId:)**

1. **设置会话的置顶状态**

**[setConversationToTop（targetId）](http://rongcloud.cn/docs/api/ios/imlib/Classes/RCIMClient.html" \l "//api/name/setConversationToTop:targetId:isTop:)**

1. **获取会话中的草稿信息**

**[getTextMessageDraft（targetId）](http://rongcloud.cn/docs/api/ios/imlib/Classes/RCIMClient.html" \l "//api/name/getTextMessageDraft:targetId:)**

1. **保存草稿信息**

**[saveTextMessageDraft(targetId）](http://rongcloud.cn/docs/api/ios/imlib/Classes/RCIMClient.html" \l "//api/name/saveTextMessageDraft:targetId:content:)**

1. **删除会话中的草稿信息**

**[clearTextMessageDraft(targetId）](http://rongcloud.cn/docs/api/ios/imlib/Classes/RCIMClient.html" \l "//api/name/clearTextMessageDraft:targetId:)**

1. **获取所有的未读消息数**

**[getTotalUnreadCount](http://rongcloud.cn/docs/api/ios/imlib/Classes/RCIMClient.html" \l "//api/name/getTotalUnreadCount)()**

1. **获取某个会话内的未读消息数**

**[getUnreadCount(targetId)](http://rongcloud.cn/docs/api/ios/imlib/Classes/RCIMClient.html" \l "//api/name/getUnreadCount:targetId:)**

1. **获取某个类型的会话中所有的未读消息数**

**[getUnreadCount(](http://rongcloud.cn/docs/api/ios/imlib/Classes/RCIMClient.html" \l "//api/name/getUnreadCount:))**

1. **清除某个会话中的未读消息数**

**[clearMessagesUnreadStatus(targetId)](http://rongcloud.cn/docs/api/ios/imlib/Classes/RCIMClient.html" \l "//api/name/clearMessagesUnreadStatus:targetId:)**

1. **设置会话的消息提醒状态**

**[setConversationNotificationStatus(targetId)](http://rongcloud.cn/docs/api/ios/imlib/Classes/RCIMClient.html" \l "//api/name/setConversationNotificationStatus:targetId:isBlocked:success:error:)**

1. **查询会话的消息提醒状态**

**[getConversationNotificationStatus(targetId](http://rongcloud.cn/docs/api/ios/imlib/Classes/RCIMClient.html" \l "//api/name/getConversationNotificationStatus:targetId:success:error:))**

1. **全局屏蔽某个时间段的消息提醒**

**[setNotificationQuietHours()](http://rongcloud.cn/docs/api/ios/imlib/Classes/RCIMClient.html" \l "//api/name/setNotificationQuietHours:spanMins:success:error:)**

1. **删除已设置的全局时间段消息提醒屏蔽**

**[removeNotificationQuietHours()](http://rongcloud.cn/docs/api/ios/imlib/Classes/RCIMClient.html" \l "//api/name/removeNotificationQuietHours:error:)**

1. **查询已设置的全局时间段消息提醒屏蔽**

**[getNotificationQuietHours()](http://rongcloud.cn/docs/api/ios/imlib/Classes/RCIMClient.html" \l "//api/name/getNotificationQuietHours:error:)**

1. **全局屏蔽某个时间段的消息提醒**

**[setConversationNotificationQuietHours(](http://rongcloud.cn/docs/api/ios/imlib/Classes/RCIMClient.html" \l "//api/name/setConversationNotificationQuietHours:spanMins:success:error:))**

1. **删除已设置的全局时间段消息提醒屏蔽**

**[removeConversationNotificationQuietHours(](http://rongcloud.cn/docs/api/ios/imlib/Classes/RCIMClient.html" \l "//api/name/removeConversationNotificationQuietHours:error:))**

1. **设置输入状态的监听器**

**[setRCTypingStatusDelegate(](http://rongcloud.cn/docs/api/ios/imlib/Classes/RCIMClient.html" \l "//api/name/setRCTypingStatusDelegate:))**

1. **向会话中发送正在输入的状态**

**[sendTypingStatus(targetId,contentType)](http://rongcloud.cn/docs/api/ios/imlib/Classes/RCIMClient.html" \l "//api/name/sendTypingStatus:targetId:contentType:)**

1. **将某个用户加入黑名单**

**[addToBlacklist](http://rongcloud.cn/docs/api/ios/imlib/Classes/RCIMClient.html" \l "//api/name/addToBlacklist:success:error:)()**

1. **将某个用户移出黑名单**

**[removeFromBlacklist](http://rongcloud.cn/docs/api/ios/imlib/Classes/RCIMClient.html" \l "//api/name/removeFromBlacklist:success:error:)()**

1. **查询某个用户是否已经在黑名单中**

**[getBlacklistStatus](http://rongcloud.cn/docs/api/ios/imlib/Classes/RCIMClient.html" \l "//api/name/getBlacklistStatus:success:error:)()**

1. **查询已经设置的黑名单列表**

**[getBlacklist()](http://rongcloud.cn/docs/api/ios/imlib/Classes/RCIMClient.html" \l "//api/name/getBlacklist:error:)**

1. **创建讨论组**

**[createDiscussion(userIdList](http://rongcloud.cn/docs/api/ios/imlib/Classes/RCIMClient.html" \l "//api/name/createDiscussion:userIdList:success:error:))**

1. **讨论组加人，将用户加入讨论组**

**[addMemberToDiscussion(userIdList](http://rongcloud.cn/docs/api/ios/imlib/Classes/RCIMClient.html" \l "//api/name/addMemberToDiscussion:userIdList:success:error:))**

1. **讨论组踢人，将用户移出讨论组**

**[removeMemberFromDiscussion(userId](http://rongcloud.cn/docs/api/ios/imlib/Classes/RCIMClient.html" \l "//api/name/removeMemberFromDiscussion:userId:success:error:))**

1. **退出当前讨论组**

**[quitDiscussion](http://rongcloud.cn/docs/api/ios/imlib/Classes/RCIMClient.html" \l "//api/name/quitDiscussion:success:error:)()**

1. **获取讨论组的信息**

**[getDiscussion](http://rongcloud.cn/docs/api/ios/imlib/Classes/RCIMClient.html" \l "//api/name/getDiscussion:success:error:)()**

1. **设置讨论组名称**

**[setDiscussionName(name](http://rongcloud.cn/docs/api/ios/imlib/Classes/RCIMClient.html" \l "//api/name/setDiscussionName:name:success:error:))**

1. **设置讨论组是否开放加人权限**

**[setDiscussionInviteStatus(isOpen](http://rongcloud.cn/docs/api/ios/imlib/Classes/RCIMClient.html" \l "//api/name/setDiscussionInviteStatus:isOpen:success:error:))**

1. **同步当前用户所在的群组列表信息**

**[syncGroups](http://rongcloud.cn/docs/api/ios/imlib/Classes/RCIMClient.html" \l "//api/name/syncGroups:success:error:)()**

1. **加入群组**

**[joinGroup(groupName](http://rongcloud.cn/docs/api/ios/imlib/Classes/RCIMClient.html" \l "//api/name/joinGroup:groupName:success:error:))**

1. **退出群组**

**[quitGroup](http://rongcloud.cn/docs/api/ios/imlib/Classes/RCIMClient.html" \l "//api/name/quitGroup:success:error:)()**

1. **加入聊天室（如果聊天室不存在则会创建）**

**[joinChatRoom:messageCount](http://rongcloud.cn/docs/api/ios/imlib/Classes/RCIMClient.html" \l "//api/name/joinChatRoom:messageCount:success:error:)()**

1. **加入已经存在的聊天室（）**

**[joinExistChatRoom(messageCount](http://rongcloud.cn/docs/api/ios/imlib/Classes/RCIMClient.html" \l "//api/name/joinExistChatRoom:messageCount:success:error:))**

1. **退出聊天室**

**[quitChatRoom](http://rongcloud.cn/docs/api/ios/imlib/Classes/RCIMClient.html" \l "//api/name/quitChatRoom:success:error:)()**

1. **获取聊天室的信息（包含部分成员信息和当前聊天室中的成员总数）**

**[getChatRoomInfo(count)](http://rongcloud.cn/docs/api/ios/imlib/Classes/RCIMClient.html" \l "//api/name/getChatRoomInfo:count:order:success:error:)**

1. **统计本地通知的事件**

**[recordLocalNotificationEvent()](http://rongcloud.cn/docs/api/ios/imlib/Classes/RCIMClient.html" \l "//api/name/recordLocalNotificationEvent:)**

1. **统计远程推送的事件**

**[recordRemoteNotificationEvent()](http://rongcloud.cn/docs/api/ios/imlib/Classes/RCIMClient.html" \l "//api/name/recordRemoteNotificationEvent:)**

1. **发起客服聊天**

**[startCustomerService(info）](http://rongcloud.cn/docs/api/ios/imlib/Classes/RCIMClient.html" \l "//api/name/startCustomerService:info:onSuccess:onError:onModeType:onPullEvaluation:onSelectGroup:onQuit:)**

1. **结束客服聊天**

**[stopCustomerService()](http://rongcloud.cn/docs/api/ios/imlib/Classes/RCIMClient.html" \l "//api/name/stopCustomerService:)**

1. **选择客服分组模式**

**[selectCustomerServiceGroup（withGroupId）](http://rongcloud.cn/docs/api/ios/imlib/Classes/RCIMClient.html" \l "//api/name/selectCustomerServiceGroup:withGroupId:)**

1. **切换客服模式**

**[switchToHumanMode()](http://rongcloud.cn/docs/api/ios/imlib/Classes/RCIMClient.html" \l "//api/name/switchToHumanMode:)**

1. **评价机器人客服，用于对单条机器人应答的评价。**

**[evaluateCustomerService（knownledgeId)](http://rongcloud.cn/docs/api/ios/imlib/Classes/RCIMClient.html" \l "//api/name/evaluateCustomerService:knownledgeId:robotValue:suggest:)**

1. **评价人工客服**

**[evaluateCustomerService(dialogId)](http://rongcloud.cn/docs/api/ios/imlib/Classes/RCIMClient.html" \l "//api/name/evaluateCustomerService:dialogId:humanValue:suggest:)**

1. **设置日志级别**

**[logLevel](http://rongcloud.cn/docs/api/ios/imlib/Classes/RCIMClient.html" \l "//api/name/logLevel)()**

1. **获取当前KFLib SDK的版本号**

**[getSDKVersion](http://rongcloud.cn/docs/api/ios/imlib/Classes/RCIMClient.html" \l "//api/name/getSDKVersion)()**

#### 消息模块类图：

**命名空间：**kflib.message

##### 透传/命令消息:CommandMessage

|  |  |  |  |
| --- | --- | --- | --- |
| **类名** | **CommandMessage（透传/命令消息）** | | |
| 方法 | 返回值 | 参数 | 描述 |
| getData() | string |  | 获取命令数据，可以为任意格式，如 JSON |
| getName() | string |  | 获取命令名 |
| obtain | CommandMessage |  | 创建消息实例 |
| setData | void |  | 设置命令数据，可以为任意格式，如 JSON |
| setName | void |  | 设置命令名 |

##### 图片消息：ImageMessage

|  |  |  |  |
| --- | --- | --- | --- |
| **类名** |  | | |
| 方法 | 返回值 | 参数 | 描述 |
| getLocalUri() | string |  | 获取本地图片地址（file:///） |
| getExtra() | string |  | 获取消息附加信息 |
| getRemoteUri() | string |  | 获取网络图片地址（http://） |
| getThumUri() | string |  | 获取缩略图Uri |
| isFull() | bool |  | 是否是原图 |
| isUpLoadExp() | bool |  | 是否上传失败 |
|  |  |  |  |

##### 文本消息：TextMessage

|  |  |  |  |
| --- | --- | --- | --- |
| **类名** |  | | |
| 方法 | 返回值 | 参数 | 描述 |
| getContent() | string |  | 获取文字消息的内容 |
| getExtra() | string |  | 获取消息扩展信息 |
| setContent() | void |  | 设置文字消息的内容 |
| setExtra() | void |  | 设置消息扩展信息 |

##### 位置消息：LocationMessage

|  |  |  |  |
| --- | --- | --- | --- |
| **类名** |  | | |
| 方法 | 返回值 | 参数 | 描述 |
| getLat() | double |  | 获取纬度 |
| getLng() | double |  | 获取经度 |
| getPoi() | double |  | 获取POI信息 |
| getExtra | string |  | 获取扩展信息 |
| set... | void |  |  |

##### 语音消息：VoiceMessage

|  |  |  |  |
| --- | --- | --- | --- |
| **类名** |  | | |
| 方法 | 返回值 | 参数 | 描述 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

##### 视频消息：VideoMessage

|  |  |  |  |
| --- | --- | --- | --- |
| **类名** |  | | |
| 方法 | 返回值 | 参数 | 描述 |
| getDuration() | int |  | 获取音频片段的时长 |
| getExtra() | string |  | 获取消息扩展信息 |
| getUri() | string |  | 获取音频文件的 Uri |
| setDuration() | void | Int | 设置音频片段的时长 |
| setExtra() | void | Int | 设置消息扩展信息 |
| setUri() | void | Int | 设置音频文件的 Uri |

##### 文件消息：FileMessage

|  |  |  |  |
| --- | --- | --- | --- |
| **类名** |  | | |
| 方法 | 返回值 | 参数 | 描述 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

##### 系统消息：SystemMessage

|  |  |  |  |
| --- | --- | --- | --- |
| **类名** |  | | |
| 方法 | 返回值 | 参数 | 描述 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

##### 自定义扩展消息：UserdefMessage

|  |  |  |  |
| --- | --- | --- | --- |
| **类名** |  | | |
| 方法 | 返回值 | 参数 | 描述 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

### KFKit.js(界面组件)

* KFKit核心类
* KFIM
* 会话列表
* [KFConversationListView](http://rongcloud.cn/docs/api/ios/imkit/Classes/RCConversationListViewController.html)
* 会话列表 Cell
* [KFConversationBaseCell](http://rongcloud.cn/docs/api/ios/imkit/Classes/RCConversationBaseCell.html)
* [KFConversationCell](http://rongcloud.cn/docs/api/ios/imkit/Classes/RCConversationCell.html)
* [KFConversationModel](http://rongcloud.cn/docs/api/ios/imkit/Classes/RCConversationModel.html)
* 聊天界面
* [KFConversationView](http://rongcloud.cn/docs/api/ios/imkit/Classes/RCConversationViewController.html)
* 输入扩展
* [KFEmojiBoardView](http://rongcloud.cn/docs/api/ios/imkit/Classes/RCEmojiBoardView.html)
* [KFPluginBoardView](http://rongcloud.cn/docs/api/ios/imkit/Classes/RCPluginBoardView.html)
* 消息 Cell
* [KFMessageBaseCell](http://rongcloud.cn/docs/api/ios/imkit/Classes/RCMessageBaseCell.html)
* [KFMessageCell](http://rongcloud.cn/docs/api/ios/imkit/Classes/RCMessageCell.html)
* [KFTextMessageCell](http://rongcloud.cn/docs/api/ios/imkit/Classes/RCTextMessageCell.html)
* [KFImageMessageCell](http://rongcloud.cn/docs/api/ios/imkit/Classes/RCImageMessageCell.html)
* [KFLocationMessageCell](http://rongcloud.cn/docs/api/ios/imkit/Classes/RCLocationMessageCell.html)
* [KFVoiceMessageCell](http://rongcloud.cn/docs/api/ios/imkit/Classes/RCVoiceMessageCell.html)
* [KFRichContentMessageCell](http://rongcloud.cn/docs/api/ios/imkit/Classes/RCRichContentMessageCell.html)
* [KFTipMessageCell](http://rongcloud.cn/docs/api/ios/imkit/Classes/RCTipMessageCell.html)
* [KFMessageModel](http://rongcloud.cn/docs/api/ios/imkit/Classes/RCMessageModel.html)
* 连接状态监听
* [KFIMConnectionStatusDelegate](http://rongcloud.cn/docs/api/ios/imkit/Protocols/RCIMConnectionStatusDelegate.html)
* 消息接收监听
* [KFIMReceiveMessageDelegate](http://rongcloud.cn/docs/api/ios/imkit/Protocols/RCIMReceiveMessageDelegate.html)
* 用户信息提供者与群组信息提供者
* [KFIMUserInfoDataSource](http://rongcloud.cn/docs/api/ios/imkit/Protocols/RCIMUserInfoDataSource.html)
* [KFIMGroupInfoDataSource](http://rongcloud.cn/docs/api/ios/imkit/Protocols/RCIMGroupInfoDataSource.html)

|  |  |  |
| --- | --- | --- |
| 类名 | 描述 | 方法成员 |
| KFKit | IM 界面组件核心类。  所有 IM 相关界面、功能都由此调用和设置。 | 方法： |
| KFKit.ConnectionStatusListener | 连接状态监听器，以获取连接相关状态。 | 方法： |
| KFKit.ConnectionStatusListener.ConnectionStatus | 连接状态枚举。 | 方法： |
| KFKit.ConversationBehaviorListener | 会话界面操作的监听器。 | 方法： |
| KFKit.GetBlacklistCallback | 获取黑名单回调。 | 方法： |
| KFKit.GetBlacklistCallback.ErrorCode | 获取黑名单错误代码枚举。 | 方法： |
| KFKit.GetFriendsProvider | 好友列表的提供者。  IMKit 本身不保存 App 的好友关系，如果在聊天中需要使用好友关系时（如：需要选择好友加入群聊），RongIM 将调用此 Provider 获取好友列表信息。 | 方法： |
| KFKit.GetGroupInfoProvider | 群组信息的提供者.  RongIM 本身不保存群组信息，如果在聊天中需要使用群组信息，RongIM 将调用此 Provider 获取群组信息。 | 方法： |
| KFKit.GetUserBlacklistCallback | 获取用户是否在黑名单回调。 | 方法： |
| KFKit.GetUserBlacklistCallback.ErrorCode | 获取黑名单错误代码枚举。 | 方法： |
| KFKit.GetUserInfoProvider | 用户信息的提供者。  如果在聊天中遇到的聊天对象是没有登录过的用户（即没有通过融云服务器鉴权过的），RongIM 是不知道用户信息的，RongIM 将调用此 Provider 获取用户信息。 | 方法： |
| KFKit.LocationProvider |  | 方法： |
| KFKit.LocationProvider.LocationCallback |  | 方法： |
| KFKit.OnReceiveMessageListener |  | 方法： |
| KFKit.OnSendMessageListener |  | 方法： |
| KFKit.OperationCallback |  | 方法： |
| KFKit.OperationCallback.ErrorCode |  | 方法： |

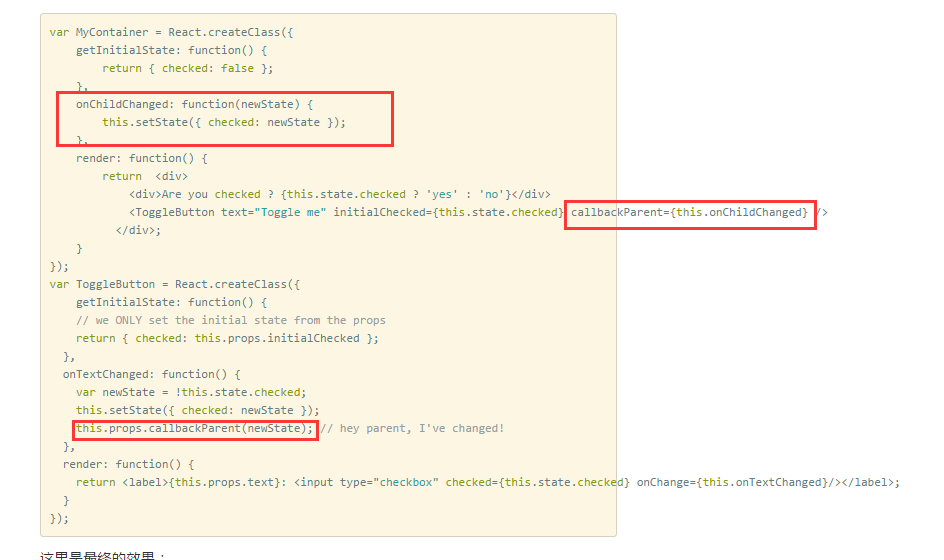
# 问题记录：

1. 加载页面白屏问题

方案：[React Server Side Rendering](https://www.google.com.hk/url?sa=t&rct=j&q=&esrc=s&source=web&cd=2&ved=0ahUKEwizyI22zbPOAhWKrI8KHV6gAN4QFggiMAE&url=https://blog.coding.net/blog/React-Server-Side-Rendering-for-SPA-SEO&usg=AFQjCNEQxQrUL8pDcKHL1LBCNVZw2rc54w" \t "https://www.google.com.hk/_blank)

1. 父子组件通信

【父组件】向【子组件】传值,【子组件】向【父组件】传值时，利用Props,state来传值，



如果组件之间嵌套太深，则不要考虑props,唯一的方式就是通过某些“事件”，一个组件去订阅这个事件，另外一个组件去触发这个事件，这是任何事件驱动的系统中最基本的两种操作：订阅或者监听一个事件，发送/触发/发布/调度这个事件去通知哪些订阅者。

介绍一种简单的模式来处理事件：

* Publish/Subscribe:类似于很多语言中的事件总线EventBus广播的方式,React中比较常用的库是PubSubJS。

第一步：命令切换到项目目录，执行

npm install pubsub-js

第二步：引入模块

Const PubSub=require(‘pubsub-js’)

第三步：订阅事件

var mySubscriber=function(msg,data){

};

var token=PubSub.subscribe(‘my topic’,mySubscriber);

第四步：发布事件

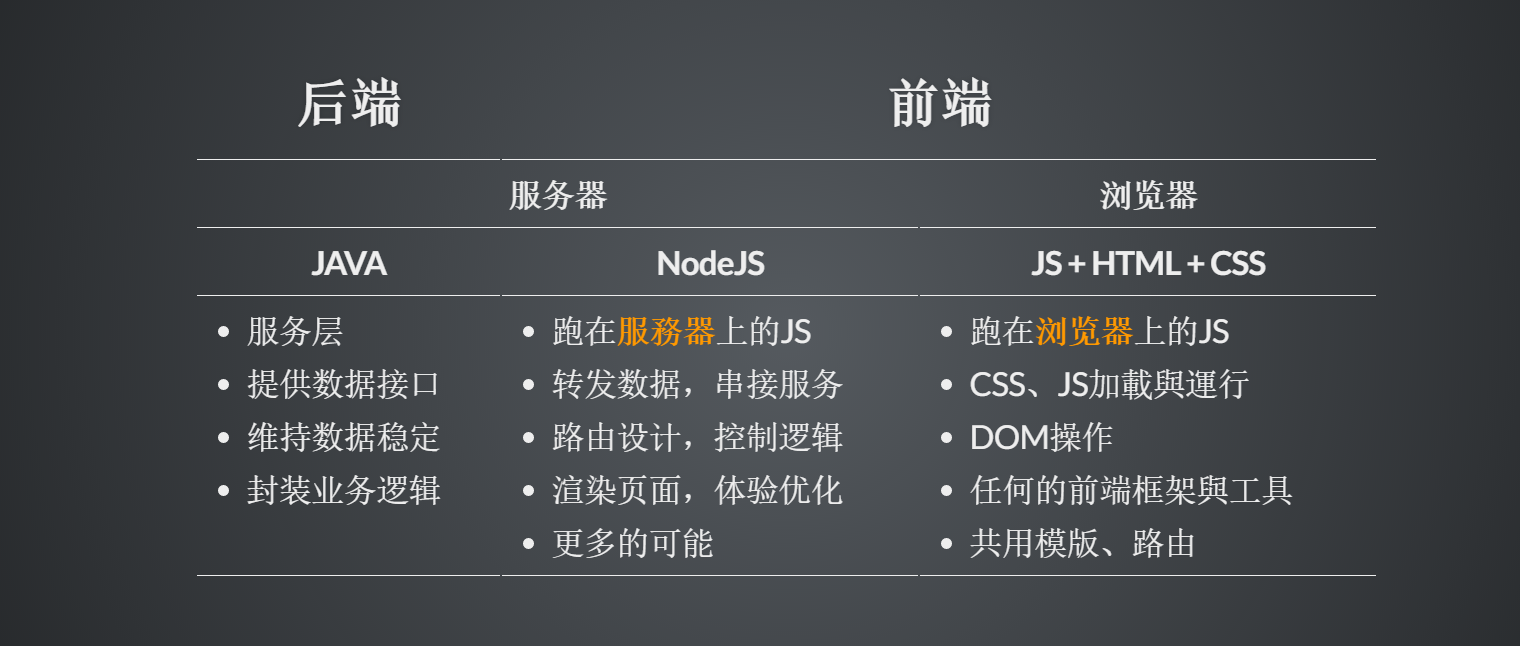
//异步发布事件，非阻塞，无需等待订阅者处理结束

PubSub.publish(‘my topic’,’hello world’);

//同步发布事件，阻塞，需要等待订阅者处理结束

PubSub.publishSync(‘my topic’,’hello world’)

1. 前后端分离中引入node.js用于和server通信



1. Redux中的 state 树太大会不会有性能问题?

其它react技术栈

| **library** | **用途** |
| --- | --- |
| font-detective | 检测系统中安装的字体列表 |
| extend | 拷贝一个对象 |
| draft-js | facebook 出品的 React 富文本编辑器，高度可定制化 |
| draft-js-export-html | 将 draft-js 的数据转换成 HTML 代码，定制化不高，有高定制化需求时，需要改代码 |
| material-design-icons | 一系列 Material 风格的 Icon，可以结合 material-ui 中的 FontIcon 使用 |
| md5-file | 计算文件 MD5，提供同步和异步两种计算方式 |
| node-native-zip | 文件的压缩和解压缩，非常好用 |
| xmlbuilder | XML 生成器，简单好用 |
| react-color | React 组件化的颜色选择器，支持各种方式的选择！极其推荐 |
| react-resizeable-and-movable | React 组件化的拖动 & 改变大小的模块 |
| qrcode.react | React 组件化的二维码生成器 |
| request | 用来下载 & 上传 |

# 后续规划:

# 注意事项：

* 合理使用css
  + 正确使用Display属性 Display属性会影响页面的渲染，因此请合理使用
  + display:inline后不应该再使用width、height、margin、padding以及float
  + display:inline-block后不应该再使用float
  + display:block后不应该再使用vertical-align
  + display:table-\*后不应该再使用margin或者float
  + 不滥用float
  + 不声明过多的font-size
  + 值为0时不需要单位
  + 标准化各种浏览器前缀
    - 无前缀应放在最后
  + CSS动画只用 （-webkit- 无前缀）两种即可
  + 其它前缀为 -webkit- -moz- -ms- 无前缀 四种，（-o-Opera浏览器改用blink内核，所以淘汰）
* 选择器
  + 避免让选择符看起来像是正则表达式。高级选择器不容易读懂，执行耗时也长
  + 尽量使用ID选择器
  + 尽量使用css3动画
* 资源加载
  + 使用srcset
  + 首次加载不超过1024KB（或者可以说是越小越好）
* html和js
  + 减少重绘和回流
  + 缓存dom选择和计算
  + 缓存列表.length
  + 尽量使用事件代理,避免批量绑定事件
  + 使用touchstart，touchend代替click
  + Html使用viewport
  + 减少dom节点
  + 合理使用requestAnimationFrame动画代替setTimeOut
  + 适当使用Canvas动画
  + TouchMove, Scroll事件会导致多次渲染