JIGUANG | 极光
数 据 改 变 世 界

JIGUANG | 极光 IT大咖说 知识分享平台

JChat 演进过程

打造一款通用的 IM UI 库

极光 IM 业务部 饶旭东

17/11/18

极光规模

25万

25万开发者

60万

60万款APP

100亿

100亿移动终端

7.5亿

月独立活跃设备

90%

90%覆盖率

如何打造一款通用的 IM UI 库

目录

Part 01

UI 库与 APP 的区别

- 用户 vs 开发者
- 重复造轮子
- 可兼容性

Part 02

架构设计

- 消息处理
- 消息类型扩展
- API 设计

Part 03

性能优化

- 缓存
- 其它

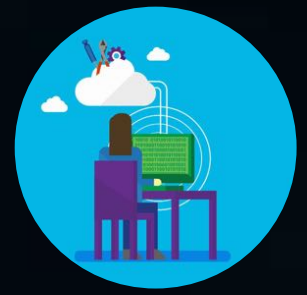
UI 库与 APP 的区别 — 用户 vs 开发者

用 户

1. 如何使用
2. 有哪些功能

VS

开发者

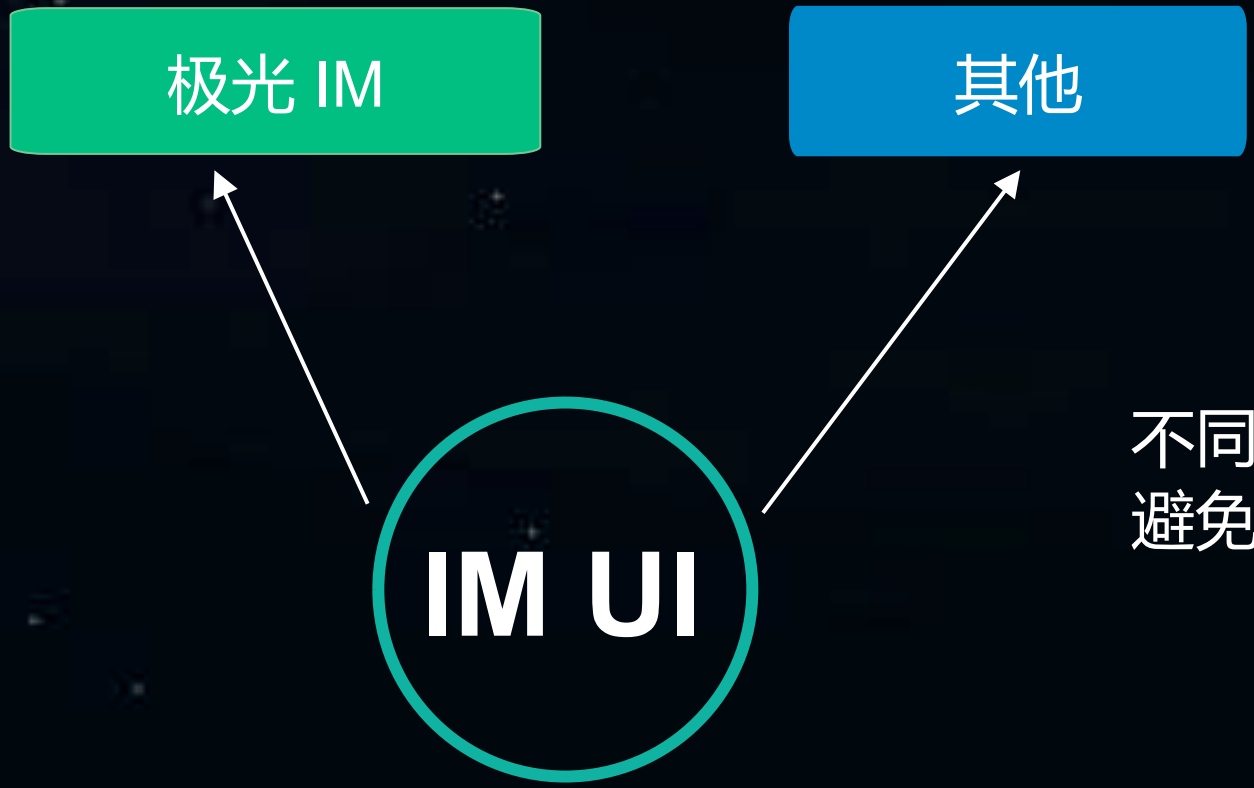


1. 有哪些功能
2. 如何实现

不要重复造轮子

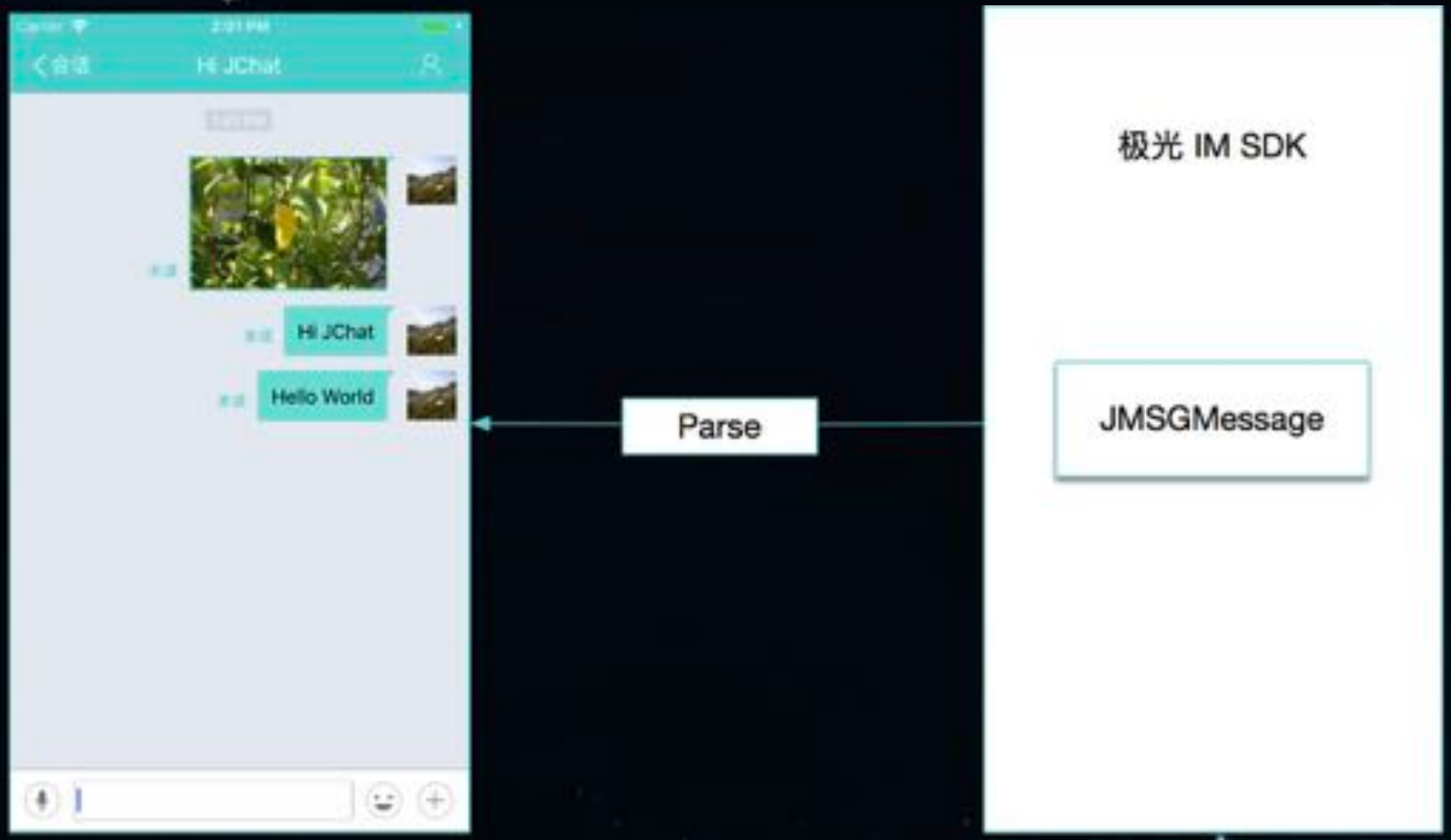
Stop Trying to Reinvent the Wheel

UI 库与 APP 的区别 — 可兼容性

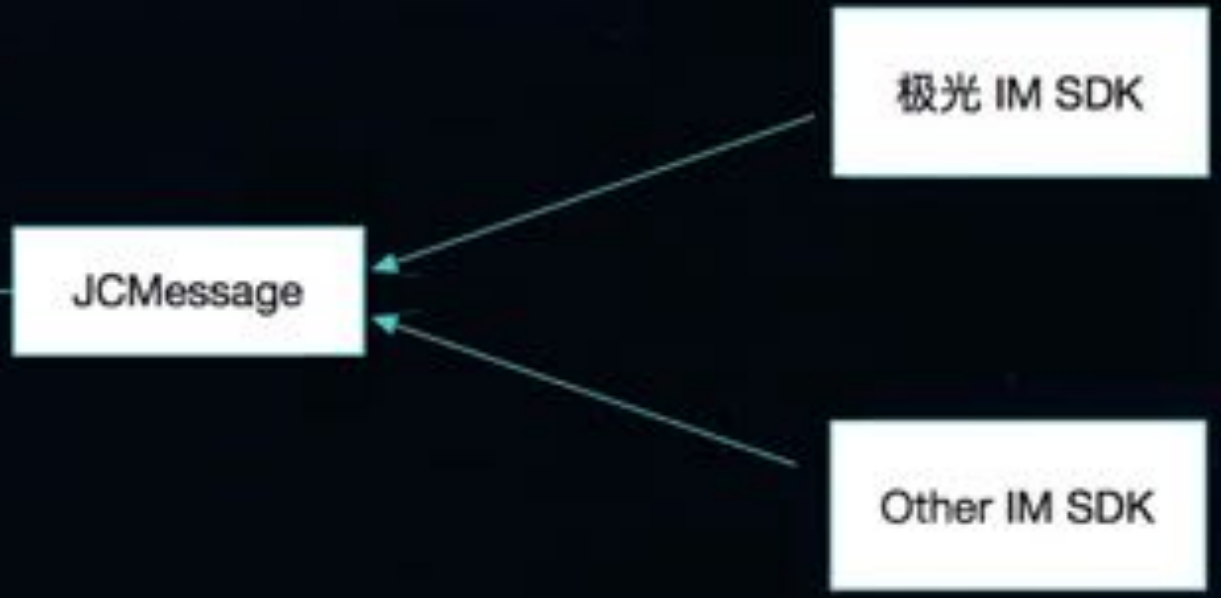


不同 IM SDK 都可以用同一套 IM UI，
避免重发开发 ，节省开发时间

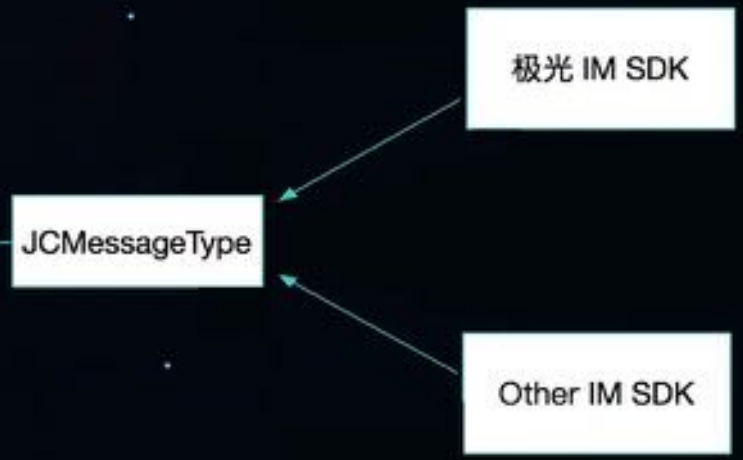
架构设计 — 消息处理(老版本)



架构设计 — 消息处理(新版本)

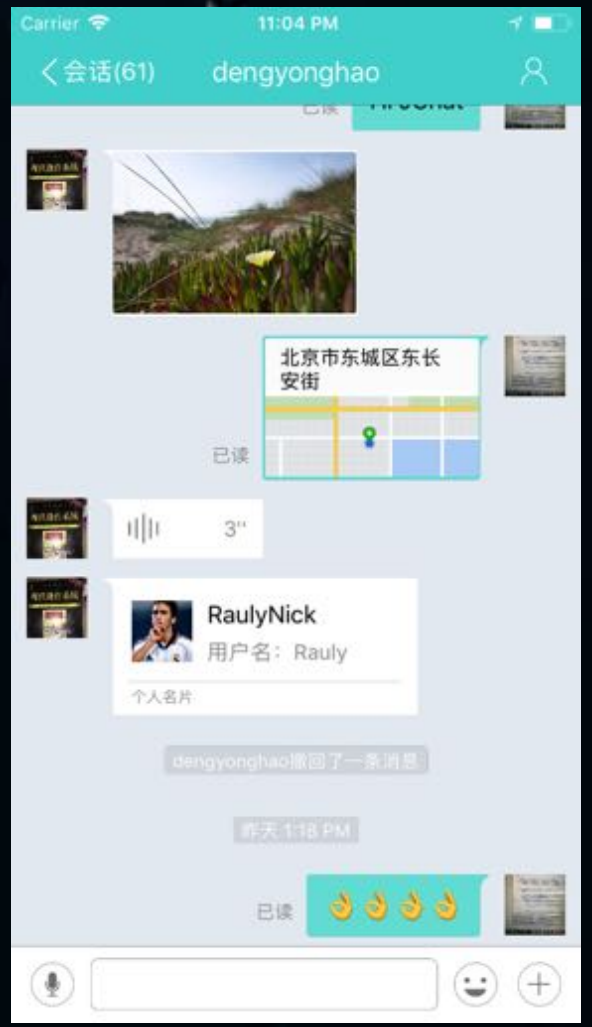


架构设计 — 消息处理




```
protocol JCMessageType: class {
    var msgId: String { get }
    var content: JCMessageContentType { get }
    var options: JCMessageOptions { get }
    var targetType: MessageTargetType { get }
    // ...
}
```

架构设计 — 消息类型的扩展



Message Type：

- Text Message
- Image Message
- Voice Message
- Video Message
- File Message
- Location Message
- ……

架构设计 — 消息类型的扩展

```
protocol JCMessageContentType: class  {
    func sizeThatFits(_ size: CGSize) -> CGSize
    static var viewType: JCMessageContentViewType.Type { get }
}
protocol JCMessageContentViewType: class {
    init()
    func apply(_ message: JCMessageType)
}
```

架构设计 — API 设计

最小化原则：

1. 尽可能少的接口来完成任务

2. 尽可能少的访问权限

架构设计 — API 设计

ChatView API :

```
public func insert(_ newMessage: JCMessageType, at index: Int)
public func append(_ newMessage: JCMessageType)
public func update(_ newMessage: JCMessageType, at index: Int)
public func removeAll()
public func remove(contentOf indexs: Array<Int>)
```

性能优化 — 缓存

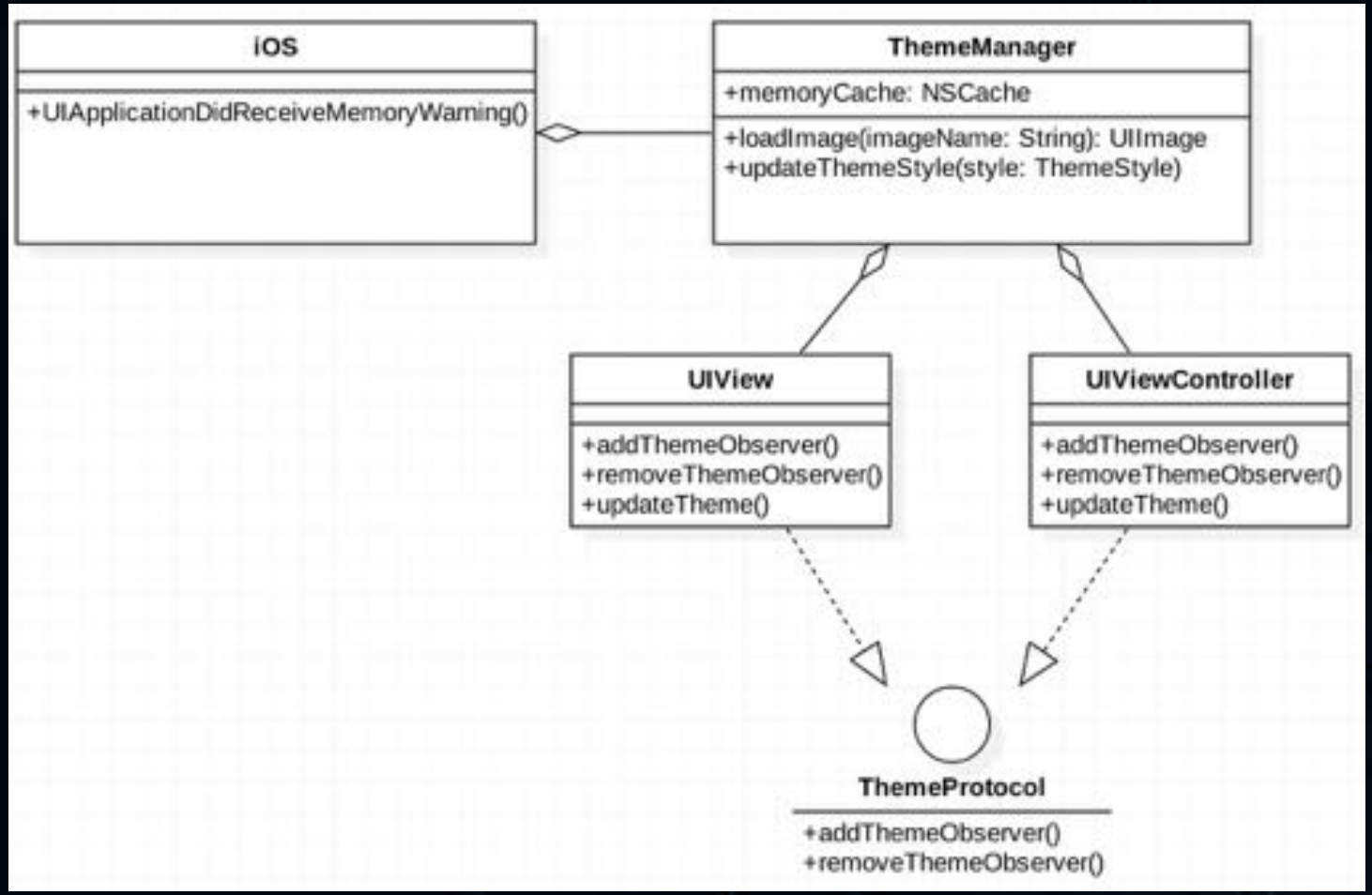
1. Cell Size 缓存

2. 网络下载资源缓存

3. 本地图片资源缓存

性能优化 — 缓存

主题管理实现：



性能优化 — 其它

1. 离屏渲染（Offscreen-Rendered）
2. 图层混合（Blended Layers）
3. 复杂界面不使用 autoLayout
4. ……

END

THANKS