聊天

SDK 提供教室内的群聊、私聊功能（不包含显示视图），可以发送文字、图片、表情三种类型的消息，提供禁言机制。

- 聊天视图：需自行创建，SDK 提供聊天管理类型 `BJLChatVM` 。
- 显示聊天的 UI 界面是需要重点优化的，优化方式可以使用高度缓存，手动计算高度，不使用自动布局等方式来优化，聊天界面一直占用主线程会导致音视频和课件不同步，界面一直卡顿等问题。如果成功优化之后依旧存在卡顿，可以使用调试工具针对性能消耗较大的功能进行针对性的优化。

聊天
群聊、私聊发送消息（文字、图片、表情）
监听收到消息
全体/个人禁止聊天
私聊消息
置顶消息
聊天翻译
聊天消息撤回
获取配置聊天快捷回复词
图文直播

1. 聊天消息监听

- 获取所有消息，SDK【不】保存聊天消息，需要上层自行维护。

```
/**
 收到历史消息 - 需要覆盖更新本地缓存的消息
 #discussion 首次进教室或断开重连会收到此回调
 #param receivedMessages 收到的所有消息
 */
[self bjl_observe:BJLMakeMethod(self.room.chatVM, receivedMessagesDidOverwrite:)
        observer:^BOOL(NSArray<BJLMessage *> *_Nullable messages) {
    bjl_strongify(self);
    if (messages.count) {
        [self.messages removeAllObjects];
        if (messages.count > 0) {
            [self.messages addObjectsFromArray:messages];
        }
        [self.tableView bjl_clearHeightCaches];
        [self.tableView reloadData];
        [self scrollToTheEndTableView];
    }
    return YES;
}];
```

- 监听消息增量更新。

```
[self bjl_observe:BJLMakeMethod(self.room.chatVM, didReceiveMessages:)
```

```
        observer:^BOOL(NSArray<BJLMessage *> *messages) {
    bjl_strongify(self);
    if (!messages.count) {
        return YES;
    }
    [self.messages addObjectsFromArray:messages];
    [self.tableView reloadData];
    return YES;
}
```

- 用户被踢出聊天服务器连接

```
/**
 收到被踢出的消息
 #param messages 消息
 */
- (BJLObservable)didReceiveKickOutMessage;

example:
    [self bjl_observe:BJLMakeMethod(self.room.chatVM, didReceiveKickOutMessage) observer:^BJLControlObserving{
        bjl_strongify(self);
        // TODO： 用户被提出聊天，此时聊天消息发送只有其他人也收不到，可以自行处理UI
        return YES;
    }];
```

2. 禁止聊天

```
/**
```

```
 老师: 设置全体禁言状态
 助教：设置自己所在组全组禁言
 #discussion 当前用户为老师或者大班助教设置成功后修改 `forbidAll`，否则若为分组助教设置成功后修改 `forbidMyGroup`
 #param forbidAll YES：全体或自己所在分组禁言，NO：取消全体/自己所在分组禁言
 #return BJLError:
 BJLErrorCode_invalidUserRole  错误权限，要求老师或助教权限
 */
- (nullable BJLError *)sendForbidAll:(BOOL)forbidAll;

/**
 全体禁言状态
 #discussion 全体或自己所在分组或个人被禁言都不能发送聊天消息，参考 `forbidMe`，`forbidMyGroup`
 */
@property (nonatomic, readonly) BOOL forbidAll;
```

```
/**
 学生: 当前用户被禁言状态
 #discussion 全体或自己所在分组或个人被禁言都不能发送聊天消息，参考 `forbidAll`，`forbidMyGroup`
 */
@property (nonatomic, readonly) BOOL forbidMe;

/**
 存在分组时, 当前用户所属的分组禁言状态
 #discussion 全体或自己所在分组或个人被禁言都不能发送聊天消息，参考 `forbidMe`，`forbidAll`
 */
@property (nonatomic, readonly) BOOL forbidMyGroup;
```

```
/**
 老师: 对某人禁言
 #discussion `duration` 为禁言时间
 #param user     禁言对象
 #param duration 禁言时长
 #return BJLError:
 BJLErrorCode_invalidUserRole   错误权限，要求老师或助教权限
 */
- (nullable BJLError *)sendForbidUser:(BJLUser *)user
                             duration:(NSTimeInterval)duration;

/**
 所有人: 收到某人被禁言通知
 #discussion `duration` 为禁言时间
 #discussion 被禁言用户可能是他人、也可能是当前用户
 #discussion 当前用户被禁言、禁言结束时会自动更新 `forbidMe`
 #param user     被禁言用户
 #param fromUser 发起禁言用户
 #param duration 禁言时长
 */
- (BJLObservable)didReceiveForbidUser:(BJLUser *)user
                             fromUser:(nullable BJLUser *)fromUser
                             duration:(NSTimeInterval)duration;
```

```
/**
```

```
 老师: 禁言用户列表
 #param forbidUserList   <userID, 剩余禁言时间>
 */
- (BJLObservable)didReceiveForbidUserList:(nullable NSDictionary<NSString *, NSNumber *> *)forbidUserList;
```

```
// 监听用户被禁言通知
[self bjl_observe:BJLMakeMethod(self.room.chatVM, didReceiveForbidUser:fromUser:duration:)
        observer:^BOOL(BJLUser *forbidUser, BJLUser *fromUser, NSTimeInterval duration){
    NSLog(@"%@被%@禁言%f秒", forbidUser.name,fromUser.name,duration);
    return YES;
}];
```

3. 发送、接收消息

- 发送消息：发送前需判断用户是否被禁言。发送聊天支持发送文字、图片、表情消息，支持文字与表情混排，3.6.0 版本后支持 GIF 表情。

```
/**
 发送文字消息，参考 `sendMessage:channel:` 方法
 #param text 消息，不能是空字符串或 nil
 #return BJLError:
 BJLErrorCode_invalidArguments  参数错误；
 BJLErrorCode_invalidCalling    错误调用，如禁言状态时调用此方法发送消息。
 */
- (nullable BJLError *)sendMessage:(NSString *)text;
```

```
/**
 指定对象，发送文字消息
 #param text 文字消息内容
 #param user 发送对象
 #return BJLError:
 BJLErrorCode_invalidArguments  参数错误；
 BJLErrorCode_invalidCalling    错误调用，如禁言状态时调用此方法发送消息。
 */
- (nullable BJLError *)sendMessage:(NSString *)text toUser:(nullable BJLUser *)user;
```

```
// 发送图片消息：image 为需要发送的图片，fileURL 为它的文件路径
bjl_weakify(self);
// 上传图片
[self.room.chatVM uploadImageFile:fileURL
                         progress:^(CGFloat progress){
    bjl_strongify(self);
    // 显示进度
    self.imageUploadingView.progress = progress;
}
                           finish:^(NSString * _Nullable imageURLString, BJLError * _Nullable error) {
    bjl_strongify(self)
    if(imageURLString){
        NSDictionary *imageData = [BJLMessage messageDataWithImageURLString:imageURLString
 imageSize:image.size];
        [self.room.chatVM sendMessageData:imageData];
    }
    else{
        NSLog(@"error:%@", error);
```

```
    }
}];
```

```
// 发送表情
// 需要在教室内才可以获取到表情，获取 emotion 数组
NSArray<BJLEmoticon *> *emoticons = [BJLEmoticon allEmoticons];
if (emoticons.count > 0) {
    // 模拟表情选择，这里直接选择第一个表情
    BJLEmoticon *emoticon = [emoticons objectAtIndex:0];
    if (emoticon) {
        //发送表情
        [self.room.chatVM sendMessageData:[BJLMessage messageDataWithEmoticonKey:emoticon.key]];
    }
}
```

- 接收消息。

收到的聊天消息为 `BJLMessage`，对其中的主要属性做以下说明：

- `ID` 作为聊天消息的唯一标识，同一场直播中是唯一的。
- `channel` 作为聊天消息的频道，可以用于区分不同聊天分组。
- `text` BJLMessageType_text消息类型的文本
- `emoticon` BJLMessageType_emoticon消息类型的表情
- `imageURLString` BJLMessageType_image消息类型的图片地址
- `toUser` 作为聊天消息的接收对象，当前消息是群聊消息值为 nil。
- `reference` 作为聊天消息的引用消息，引用消息不嵌套多层。

- `type` 聊天消息类型，表情和文字混排的消息认为是文字消息。
- `translation` 聊天翻译的结果，只有翻译后的消息才会有值。
- `sendToGroupID` 发送给某个组的消息
- `rewardType` 礼物打赏消息类型
- `fromType, toType` 聊天翻译的源语言和目标语言

```
// 接收表情和文字混排的消息，获取 NSAttributedString 显示即可
self.messageLabel.attributedText = [message attributedEmoticonStringWithEmoticonSize:fontSize

attributes:attributes

cached:YES

cachedKey:cachedKey];
```

```
// 接收文字和图片参考发送消息
// 接收表情消息 type == BJLMessageType_emoticon, 优先判断cachedImage
if (self.message.type == BJLMessageType_emoticon) {
    if (self.message.emoticon.cachedImage) {
        self.emoticonImageView.image = self.message.emoticon.cachedImage;
    }
    else {
        bjl_weakify(self);
        [self.emoticonImageView bjl_setImageWithURL:self.message.emoticon.urlStri
                                        placeholder:nil
                                         completion:^(UIImage * _Nullable image, NSError * _Nullable error,
```

```
NSURL * _Nullable imageURL) {
        bjl_strongify(self);
        if (image) {
            self.message.emoticon.cachedImage = image;
        }
    }];
  }
}
```

- 私聊、频道聊天。

```
/**
 指定对象，发送文字消息
 #param text 文字消息内容
 #param user 发送对象
 #return BJLError:
 BJLErrorCode_invalidArguments 参数错误；
 BJLErrorCode_invalidCalling 错误调用，如禁言状态时调用此方法发送消息。
 */
[self.room.chatVM sendMessage:message toUser:targetUser];

/**
 指定频道，发送文字消息
 #discussion 最多 BJLTextMaxLength_chat 个字符
 #discussion 成功后会收到消息通知
 #discussion 学生在禁言状态不能发送消息，参考 `forbidMe`、`forbidAll`
 #discussion 参考 `BJLMessage`
 #param text 消息，不能是空字符串或 nil
 #param channel 频道
 */
```

```
[self.room.chatVM sendMessage:message channel:@"小组频道"];
```

4. 私聊

```
/**
 请求私聊的历史消息
 #param user 私聊对象
 #param page  起始消息的页码 (从0开始, 每页20条)
 */
- (nullable BJLError *)loadWhisperMessagesWithTargetUser:(BJLUser *)user page:(NSUInteger)page;

/**
 收到私聊消息
 #param messages 消息
 */
- (BJLObservable)didReceiveWhisperMessages:(NSArray<BJLMessage *> *)messages targetUserNumber:(NSString *)targetUserNumber hasMore:(BOOL)hasMore;
```

5. 置顶聊天

将聊天消息置顶，取消置顶。

```
/** 第一条聊天的置顶消息 */
@property (nonatomic, readonly, nullable) BJLMessage *stickyMessage;

/** 聊天的所有置顶消息 */
@property (nonatomic, readonly, nullable) NSArray<BJLMessage *> *stickyMessageList;
```

```
6.
7. /**
8.  获取置顶消息
9.  #discussion 连接直播间后、掉线重新连接后自动调用加载
10.  #discussion 获取成功后修改 `stickyMessageList`
11.  */
12. - (void)loadStickyMessage;
13.
14. /**
15.  老师: 设置置顶消息
16.  #discussion message 为空表示取消置顶内容, 不为空表示设置新的置顶消息
17.  #discussion 设置成功后修改 `stickyMessage`
18.  #return BJLError:
19.  BJLErrorCode_invalidUserRole  错误权限，要求老师或助教权限。
20.  */
21. - (nullable BJLError *)sendStickyMessage:(BJLMessage *)message;
22.
23. /**
24.  老师: 取消某一条置顶的消息
25.  #discussion 设置成功后修改 `stickyMessageList`,
26.  @param message messgae
27.  BJLErrorCode_invalidUserRole  错误权限，要求老师或助教权限。
28.  */
29. - (nullable BJLError *)cancelStickyMessage:(BJLMessage *)message;
```

6. 聊天翻译

支持聊天消息的英汉互译。

```objc
/** 语种列表 */
typedef NS_ENUM(NSInteger, BJLMessageLanguageType) {
    /** 未知语言 */
    BJLMessageLanguageType_NONE,
    /** 中文简体 */
    BJLMessageLanguageType_ZH,
    /** 中文繁体 */
    BJLMessageLanguageType_CHT,
    /** 英语 */
    BJLMessageLanguageType_EN,
    /** 日语 */
    BJLMessageLanguageType_JP,
    /** 越南语 */
    BJLMessageLanguageType_VIE,
    /** 印尼语 */
    BJLMessageLanguageType_ID,
    /** 柬埔寨语 */
    BJLMessageLanguageType_HKM
};
```

```objc

/** 默认翻译目标语言 */
@property (nonatomic, readonly) BJLMessageLanguageType defaultTranslateToLanguageType;

/**
 翻译聊天消息，每秒最多仅允许发送一条，翻译源语言类型自动检测
 #param message 消息
 #param messageUUID 用于区分消息的唯一ID,不建议使用BJLMessage.ID, 不保证其唯一性
```

```
 #param BJLMessageLanguageType 目标翻译语言类型
 */
- (nullable BJLError *)translateMessage:(BJLMessage *)message
                            messageUUID:(nullable NSString *)messageUUID
                       targetLanguageType:(BJLMessageLanguageType)languageType;

/**
 收到消息翻译的结果
 #param translation 消息翻译的结果
 #param messageUUID 消息唯一ID
 #param from 消息翻译源语言类型
 #param to 消息翻译目标语言类型
 */
- (BJLObservable)didReceiveMessageTranslation:(NSString *)translation
                                  messageUUID:(nullable NSString *)messageUUID
                                         from:(BJLMessageLanguageType)from
                                           to:(BJLMessageLanguageType)to;
```

7. 聊天撤回

支持撤回任意时间，任意内容的聊天消息，撤回消息后，需要 UI 自行处理消息的展示。

```
/**
 撤回消息
 #discussion 老师可以撤回任意用户消息，学生可以撤回自己消息
```

```
 #param BJLMessage 聊天消息
 #return error
 */
- (nullable BJLError *)revokeMessage:(BJLMessage *)message;

/**
 撤回消息成功
 #param messageID 消息ID
 #param isCurrentUserRevoke 是否是当前用户撤回的消息
 */
- (BJLObservable)didRevokeMessageWithID:(NSString *)messageID isCurrentUserRevoke:(BOOL)isCurrentUserRevoke;
```

8. 聊天快捷回复

```
// 可以获取到后台配置的聊天消息快捷回复词
- (nullable BJLError *)getQuickReplyWordsWithCompletion:(nullable void (^)(NSArray <NSString *> * _Nullable quickReplyWords, BJLError * _Nullable error))completion;
```

9. 图文直播功能

```

/** 获取图文直播配置信息 */
- (nullable BJLError *)getPicAndTextPresenterInfoWithCompletion:(nullable void (^)(NSDictionary *_Nullable dic, BJLError *_Nullable error))completion;
```

```
4.
5. /** 获取图文直播历史信息 */
6. - (nullable BJLError
   *)getPicAndTextListWithCompletion:(nullable void
   (^)(NSArray<BJLPicAndTextModel *> *_Nullable
   list, NSInteger page, BJLError *_Nullable
   error))completion;
7.
8. /** 图文直播配置更新 */
9. -
   (BJLObservable)didReceivePicAndTextPresenterInfol
   (NSDictionary *)userInfo;
10.
11. /** 收到新的图文消息*/
12. - (BJLObservable)didReceiveNewPicAndTextInfo:
   (BJLPicAndTextModel *)newPicAndTextInfo;
13.
14. /** 撤回图文消息 */
15. - (BJLObservable)didReceivePicAndTextInfoDelete:
   (NSString *)graphicID;
```

下载为pdf格式