教室工具

教室工具主要由 `BJLRoomVM` 管理，记录教室的信息、状态，以及教室的一些工具、功能，文档下方除了特别指明的功能，都是在 `BJLRoomVM` 中。可以根据实际需要选取使用。

教室工具
录课、转码、生成回放
进入教室时间，上课时间，排课时间
助教上麦、画笔、文档、公告、上下课、禁言、踢人、 云端录制权限管理
发布，获取教室公告，监听公告变化
跑马灯内容，样式
课前问卷、课后评价
点名时长，收到点名，答到
专注度检测，不专注提醒
点赞发送、收取，点赞统计
红包雨创建、发布、收取、抢红包，获取抢红包结果， 红包排行榜
测验新增、删除、加载、发布、结束、提交，获取测验列表
问答创建，发布，取消发布、回复，禁止、允许提问， 加载问答数据

网页同步打开、关闭
课后学情数据
计时器发布、暂停、结束
抢答器发布、收到、抢答、关闭、撤销
答题器发布、结束、提交、关闭、撤销，获取用户答题数据
参与标准抽奖和口令抽奖，中奖后提交联系方式
转播其他教室的音视频、课件、画笔
教室布局切换
教室积分获取、列表数据

1. 录课

录课是将当前教室的情景、信息以及互动记录录制到云端生成回放，通过回放功能可以再现教室的情景。本 SDK 不包含回放功能，如需集成请参考 iOS 点播回放 Core SDK。

录课管理类型为 `BJLServerRecordingVM`，实例 `serverRecordingVM` 在创建教室时被初始化。

- 获取云端录课状态。

```
// 云端录课状态，反映当前云端录课是否开启
BOOL serverRecording = self.room.serverRecordingVM.serverRecording;

/* 云端录制详细状态
    BJLServerRecordingState_ready,                  // 未开启云端录制
    BJLServerRecordingState_recording,              // 开启过云端录制并且未转码或者正在云端录制中，可继
```

```
续录制。如果现在在录制中，长期课可以停止录制，请求转码后开启新的录制
    BJLServerRecordingState_transcoding,
// 云端录制转码中，只能开启新的云端录制，不能继续录制。短期课不会有这个状态
    BJLServerRecordingState_disable,
// 云端录制不可用，是短期课已经录制过。长期课不会有这个状态
*/
@property (nonatomic, readonly)
BJLServerRecordingState state;
```

- 老师开启/停止云端录课：上课状态才能开启录课，参考 `roomVM.liveStarted`，此方法需要发起网络请求、检查云端录课是否可用。

```
[self.room.serverRecordingVM
requestServerRecording:YES]; // YES：开启，NO：关闭
```

```
/**
 请求当前云端录制转码状态
 #param completion 状态更新完成，可以根据状态来进行云端录制
 BJLErrorCode_invalidCalling  错误调用，如当前直播间录制类型不是云端录制；
 */
- (nullable BJLError *)requestServerRecordState:
(void (^ __nullable)(BOOL success))completion;
```

- 请求立即转码回放。大班课支持配置下课立即生成回放，参考 `shouldGeneratePlaybackAfterClass`。

```
/**
```

```
 请求立刻转码回放
 #return BJLError:
 BJLErrorCode_invalidUserRole  错误权限，要求老师或助教权限。
 */
BJLError *error = [self.room.serverRecordingVM requestServerRecordingTranscode];

// example：监听转码回放请求被接受
[self bjl_observe:BJLMakeMethod(self.room.serverRecord requestServerRecordingTranscodeAccept)
        observer:^BOOL{
    // bjl_strongify(self);
    return NO;
}];
```

- 监听云端录课不可用的通知。

```
// example：监听云端录课不可用的通知
[self bjl_observe:BJLMakeMethod(self.room.serverRecord requestServerRecordingDidFailed:)
        observer:^BOOL(NSString *message) {
    NSLog(@"request server recording failed:%@", message);
    return YES;
}];
```

2. 教室上下课

可以监听上课时间，进入教室时间，上下课状态等信息。

```
/** 上课状态 */
```

```
@property (nonatomic, readonly) BOOL liveStarted;
/** 实际上课开始时间 */
@property (nonatomic, readonly) NSTimeInterval classStartTimeMillisecond;
/** 理论排课开始时间 */
@property (nonatomic, readonly) NSTimeInterval classStartTimesecond;
/** 进入教室时间 */
@property (nonatomic, readonly) NSTimeInterval enteringTimeInterval; // seconds since 1970
```

3. 助教权限

可以获取助教拥有的教室权限，助教的权限可以由老师控制。主要的助教权限有上麦、画笔、文档、公告、上下课、禁言、踢人、云端录制。

```
// example: 监听权限变更，获取当前是否拥有上课权限
[self bjl_observe:BJLMakeMethod(self.room.roomVM, didReceiveAssistantaAuthorityChanged) observer:^BOOL{
    bjl_strongify(self);
    BOOL enableLiveStart = [self.room.roomVM getAssistantaAuthorityWithClassStartEnd];
    return YES;
}];
```

4. 公告

用户可以查看教室内由老师发布的公告，公告可包含跳转链接。 `BJLRoomVM` 提供公告的获取、发布方法，也可以监听公告变化，即时更新。

- 获取教室公告。

```
/**
 获取教室公告
 #discussion 连接教室后、掉线重新连接后自动调用加载
 #discussion 获取成功后修改 `notice`
 */
- (void)loadNotice;

/** 教室公告 */
@property (nonatomic, readonly, copy, nullable) BJLNotice *notice;
```

```
/** 大班课教室内主屏置顶公告 */
@property (nonatomic, readonly, copy, nullable) BJLMajorNotice *majorNotice;
```

- 监听公告变化，即时显示。

```
// example: 监听公告变化，即时显示
bjl_weakify(self);
[self bjl_kvo:BJLMakeProperty(self.room.roomVM, notice)
    observer:^BOOL(BJLNotice * _Nullable notice, id _Nullable oldValue, BJLPropertyChange * _Nullable change) {
    bjl_strongify(self);
    self.noticeTextView.text = notice.noticeText.length ? notice.noticeText : nil;
    return YES;
}];
```

- 发布公告。

```
// noticeText：公告内容 linkURL：公告跳转链接
[self.room.roomVM
sendNoticeWithText:noticeText
linkURL:noticeURL];
```

5. 跑马灯

跑马灯是直播间内漂浮的文字信息，可以通过后台自定义样式，内容，显示模式等。

```
/** 跑马灯内容 <包括文本内容，文本颜色，字体大小，透明度> */
@property (nonatomic, readonly, copy, nullable) BJLLamp *lamp;
```

6. 课前问卷、课后评价

课前问卷是进入教室前对用户身份信息的调查，可以配置开启。目前课后评价的使用嵌入的 webview 实现，可以获取到配置的问卷 URL 加载展示，可参考 UI SDK 的效果。

```
/**
 获取课前问卷
 #param completion 回调, isNeedFill: 是否需要填写问卷; questionURL: 问卷链接;
 */
- (void)getQuestionNaireWithCompletion:(nullable void (^)(BOOL isNeedFill, NSString *questionURL))completion;
```

课后评价是课程结束时，一个对课堂情况的可自定义的问卷反馈。目前课后评价的使用嵌入的 webview 实现，可以通过 `evaluationRequest` 获取当前教室课后评价的地址展示，具

体展示方式可参考 UI SDK 的效果。支持后台配置课后评价的内容。

```
// example：获取课后评价，webview 加载请求
self.request = [self.room.roomVM evaluationRequest];
[self.webView loadRequest:self.request];
```

7. 点名

点名由老师发布，学生接收并答到。

- 老师：发起点名。

```
/** 老师: 发起点名
 学生需要在规定时间内 `timeout` 答到
 参考 `rollcallTimeRemaining`
 */
- (nullable BJLError *)sendRollcallWithTimeout:(NSTimeInterval)timeout;
```

- 学生/老师:收到点名。

```
/**
 学生/老师: 收到点名
 #discussion 学生需要在规定时间内 `timeout` 答到 - 调用 `answerToRollcall`
 #discussion 参考 `rollcallTimeRemaining`
 #discussion 老师自己发起点名时，也会收到该回调
 #param timeout 超时时间
 */
- (BJLObservable)didReceiveRollcallWithTimeout:(NSTimeInterval)timeout;
```

- 点名倒计时。

```
/**
 点名倒计时
 #discussion 每秒更新
 */
@property (nonatomic, readonly) NSTimeInterval rollcallTimeRemaining;
```

- 学生/老师: 收到点名取消。

```
/**
 学生/老师: 收到点名取消
 #discussion 可能是老师取消、或者倒计时结束
 #discussion 参考 `rollcallTimeRemaining`
 */
- (BJLObservable)rollcallDidFinish;
```

- 答到。

```
/**
 学生: 答到
 #return BJLError:
 BJLErrorCode_invalidCalling    错误调用，如老师没有点名或者点名已过期;
 BJLErrorCode_invalidUserRole    错误权限，要求非试听学生权限。
 */
- (nullable BJLError *)answerToRollcall;
```

8. 专注度检测

SDK 会自动进行专注度检测，在后台或者 APP 处于不活跃的状态都将会认为是不专注的，老师提醒当前用户不专注时，在 APP

重新处于活跃状态时可以收到老师的提醒。

```
// example：收到不专注提醒，弹出提示
[self bjl_observe:BJLMakeMethod(self.room.roomVM, didReceiveAttentionWarning:)
        observer:^BOOL(NSString *content) {
    bjl_strongify(self);
    [self showProgressHUDWithText:content];
    return YES;
}];
```

9. 点赞

- 助教和老师可以给学生点赞，学生无法点赞，助教和老师不能被点赞，下课【不】清空点赞记录。

```
/**
 点赞数据
 #discussion key --> userNumber
 #discussion value --> 点赞数
 */
@property (nonatomic, readonly, nullable) NSDictionary<NSString *, NSNumber *> *likeList;
@property (nonatomic, readonly, nullable) NSDictionary<NSNumber *, NSNumber *> *grouplikeList;
/**
 个人点赞数据-多种奖励方式
 #discussion key --> userNumber
 #discussion value --> dic, { "key" : "key对应类型的奖励数目" }
*/
@property (nonatomic, readonly, nullable) NSDictionary<NSString *, NSDictionary *>
```

```
*mutableAwardsInfo;
```

- 点赞用户。支持多种奖励方式，后台可以配置奖励的图片样式，参考 `BJLAward` 。

```
/**
 多种点赞方式
 #param userNumber userNumber
 #param key 多种奖励方式对应的key
 #return error:
 BJLErrorCode_invalidCalling    错误调用，如用户不在线；
 BJLErrorCode_invalidUserRole    错误权限，如点赞用户不能是学生，被点赞用户不能是老师，助教。
 */
- (nullable BJLError *)sendLikeForUserNumber:(NSString *)userNumber key:(nullable NSString *)key;
```

- 收到点赞。

```
/**
 收到个人点赞
 #discussion 收到的所有点赞都在点赞记录中，包括本次收到的点赞
 #param userNumber userNumber
 #param records 点赞记录 key --> userNumber，value --> 点赞数
 */
[self bjl_observe:BJLMakeMethod(self.room.roomVM, didReceiveLikeForUserNumber:records:)
        observer:^BOOL(NSString *userNumber, NSDictionary<NSString *, NSNumber *> *records) {
```

```
    return YES;
}];
/**
 收到一次分组/全员点赞
 #discussion groupID 为 0 表示全员点赞,非0表示分组点赞，非 0 时 groupName 未分组名
 #param groupid=0 时， 为全员点赞， 不计入个人/分组点赞数量
 */
[self bjl_observe:BJLMakeMethod(self.room.roomVM, didReceiveLikeForGroupID:groupName:) observer:^BOOL(NSInteger groupID, NSString *groupName) {
    bjl_strongify(self);
    if (groupID != 0 && self.user.groupID == groupID) {
        [self updateLikeCount];
    }
    return YES;
}];
```

10. 红包雨

红包雨功能用于活跃教室互动气氛，SDK 提供红包雨的发起、结束、抢红包、最终结果、排行榜的请求和回调，UI 可以根据接口实现红包雨的效果，可以参考 UI SDK 相关模块实现。红包雨的内容是虚拟积分的形式，不使用实际的货币。

- 创建红包雨。

```
/**
 创建红包雨
 #param amount 红包总数
 #param score 学分总数
```

```
 #param duration 红包雨时长
 #param completion 红包雨活动ID
 #return task
 */
- (nullable NSURLSessionDataTask *)createEnvelopeRainWithAmount:(NSInteger)amount score:(NSInteger)score duration:(NSInteger)duration completion:(nullable void (^)(NSInteger envelopeID, BJLError * _Nullable error))completion;
```

- 抢红包。

```
/**
 抢红包
 #discussion 活动结束时，completion 也返回 0，并且没有 task
 #param envelopeID 红包雨活动ID
 #param completion 抢到的学分
 #return task
 */
- (nullable NSURLSessionDataTask *)grapEnvelopeWithID:(NSInteger)envelopeID completion:(nullable void (^)(NSInteger score, BJLError * _Nullable error))completion;
```

- 获取学生抢到的学分总数。

```
/**
 获取学生抢到的学分总数
 #param userNumber user number
 #param completion 学分总数
 #return task
 */
```

```
- (nullable NSURLSessionDataTask *)requestTotalScoreWithUserNumber:(NSString *)userNumber completion:(nullable void (^)(NSInteger totalScore, BJLError * _Nullable error))completion;
```

- 获取指定红包雨活动的最终结果。

```
/**
 获取指定红包雨活动的最终结果
 #param envelopeID 红包雨活动ID
 #param completion completion 红包雨活动结果
 #return task
 */
- (nullable NSURLSessionDataTask *)requestResultWithEnvelopeID:(NSInteger)envelopeID completion:(nullable void (^)(BJLEnvelopeResult * _Nullable result, BJLError * _Nullable error))completion;
```

- 获取指定红包雨活动的排行榜。

```
/**
 获取指定红包雨活动的排行榜
 #param envelopeID envelopeID
 #param completion completion 排行榜数据
 #return task
 */
- (nullable NSURLSessionDataTask *)requestRankListWithEnvelopeID:(NSInteger)envelopeID completion:(nullable void (^)(NSArray<BJLEnvelopeRank *> * _Nullable, BJLError * _Nullable))completion;
```

- 开始红包雨。

```
/**
 开始红包雨
 #param envelopeID 红包雨活动ID
 #param duration 红包雨时长
 #return error
 */
- (nullable BJLError *)startEnvelopRainWithID:(NSInteger)envelopeID duration:(NSInteger)duration;
```

- 收到红包雨。

```
/**
 收到红包雨
 #param envelopeID 红包雨活动ID
 #param duration 红包雨时长
 */
- (BJLObservable)didStartEnvelopRainWithID:(NSInteger)envelopeID duration:(NSInteger)duration;
```

11. 测验

测验由老师发布，学生接收并答题。每一场测验都是一个 `BJLQuiz` 对象，题目类型为 `BJLQuizQuestion`，测验的题目有 4 种类型，参考 `BJLQuizQuestionType`，分别是 单选、多选、简答、判断题，选择题和判断题会有 `BJLQuizOption` 类型的数组作为选项，简答题可以设置参考答案。测验相关的 API 参考 `BJLRoomVM` 的 `测验 V2 native 方式` 部分。

- 测验状态列表。

```
/**
 当前的新版测验状态列表
 #discussion key -> 测验 ID，value -> 测验状态（参考 `BJLQuizState`），测验的先后根据测验 ID 从小到大对应
 */
@property (nonatomic, readonly, nullable) NSDictionary<NSString *, NSNumber *> *quizStateList;
```

- 当前正在进行的测验 ID。

```
/**
 正在进行的测验
 #discussion 至多只有一个测验正在进行，如果没有正在进行的测验，值为空
 */
@property (nonatomic, readonly, nullable) NSString *currentQuizID;
```

- 加载测验列表请求。

```
/**
 加载测验列表
 #param completion quizList 仅 quiz ID，title，state 可用
 #return task
 */
[self.room.roomVM loadQuizListWithCompletion:^(NSArray<BJLQuiz *> * _Nullable quizList, BJLError * _Nullable error) {
  // bjl_strongify(self);
  // your code
```

```
}];
```

- 老师、助教：新增或更新测验。

```
/**
 新建，更新测验，老师或助教身份
 #discussion 创建新测验，测验 ID 使用 0，否则更新测验
 #param quiz 测验内容，参考 `BJLQuiz`
 #param completion 测验 ID
 #return task
 */
[self.room.roomVM updateQuiz:myQuiz
                  completion:^(NSString * _Nullable quizID, BJLError * _Nullable error) {
    // bjl_strongify(self);
    // your code
}];
```

- 老师、助教：删除测验。

```
/**
 删除测验，老师或助教身份
 #param quizID 测验 ID
 #param completion 测验 ID
 #return task
 */
[self.room.roomVM deleteQuizWithID:targetQuizID
                        completion:^(NSString * _Nullable quizID, BJLError * _Nullable error) {
    // bjl_strongify(self);
    // your code
}];
```

- 加载测验的详细内容。

```
/**
 加载测验详细内容
 #discussion 所有角色，对于老师或助教，如果测验结束了，有答题情况了，会返回测验的答题情况，对于学生，不会返回答题情况
 #param quizID 测验 ID
 #param completion BJLQuiz，测验详细信息
 #return task
 */
- (nullable NSURLSessionDataTask *)loadQuizDetailWithID:(NSString *)quizID completion:(nullable void (^)(BJLQuiz * _Nullable quiz, BJLError * _Nullable error))completion;
```

- 开始测验。

```
/**
 开始测验，老师或助教身份
 #param quizID quizID
 #param force 是否强制参加
 #return BJLError
 */
- (nullable BJLError *)startQuizWithID:(NSString *)quizID force:(BOOL)force;

// example：监听测验开始
[self bjl_observe:BJLMakeMethod(self.room.roomVM, didStartQuizWithID:force:)
        observer:(BJLMethodObserver)^BOOL(NSString *quizID, BOOL force) {
    // bjl_strongify(self);
```

```
    NSLog(@"测验%@开始，强制参加：%@", quizID, force? @"YES" : @"NO");
    return YES;
}];
```

- 结束测验。

```
/**
 老师、助教：结束测验
 #param quizID 测验 ID
 #return BJLError
 */
- (nullable BJLError *)endQuizWithID:(NSString *)quizID;

// example：监听测验结束
[self bjl_observe:BJLMakeMethod(self.room.roomVM, didEndQuizWithID:)
         observer:^BOOL(NSString *quizID) {
             // bjl_strongify(self);
             NSLog(@"测验%@结束", quizID);
             return YES;
         }];
```

- 发布测验答案。

```
/**
 发布测验答案，老师或助教身份
 #param quiz 需要发布答案的测验 ID
 #return BJLError
 */
- (nullable BJLError *)publishQuizSolutionWithID:(NSString *)quizID;
```

```
// 发布测验答案通知，可通过 KVO 监听
- (BJLObservable)didReceiveQuizWithID:(NSString *)quizID solution:(NSDictionary<NSString *, id> *)solutions;
```

- 加载当前测验。

```
/**
 加载当前测验
 #discussion 学生身份如果回答过，将返回回答的结果
 #return BJLError
 */
- (nullable BJLError *)loadCurrentQuiz;

// 测验加载回调，学生用户 如果回答过，将返回回答的结果
- (BJLObservable)didLoadCurrentQuiz:(BJLQuiz *)quiz;
```

- 提交测验。

```
/**
 提交测验，学生身份，此方法无提交完成的回调
 #param quizID 测验 ID
 #param solutions 测验回答 key -> 问题 ID，value -> 问题回答，对于 value，Radio 类型的问题的值为选项 ID，Checkbox 类型的问题的值为选项 ID 数组，ShortAnswer 类型的问题值为简答的关键内容
 #return BJLError
 */
- (nullable BJLError *)submitQuizWithID:(NSString *)quizID solution:(NSDictionary<NSString *, id> *)solutions;
```

```
/**
 收到学生提交测验，老师或助教身份
 #param quizID 测验ID
 #param solutions 学生答题情况
 */
- (BJLObservable)didSubmitQuizWithID:(NSString *)quizID solution:(NSDictionary<NSString *, id> *)solutions;
```

- 大小班场景测验。

```
/**
 位于小班，加载大班的已经结束的测验列表
 @return BJLError
 */
- (nullable BJLError *)loadParentRoomFinishedQuizList;

/**
 位于小班，收到大班结束的测验信息
 @param quizList 测验列表
 */
- (BJLObservable)didLoadParentRoomFinishedQuizList:(nullable NSArray<BJLQuiz *> *)quizList;
```

12. 问答

问答是类似于聊天的互动功能，学生提出问题，老师或者助教进行回答，问答如果在未发布状态是只有提问者和老师、助教可见，在发布状态所有人可见。问答通过分页加载。

- 获取问答数据。

```
/**
 加载指定页码的问答
 #param page 页码，从0开始计数
 #param count 每一页最多返回5条，如果数据较多，需要分页请求
 #return error
 */
BJLError *error = [self.room.roomVM loadQuestionHistoryWithPage:self.currentQuestionP
 countPerPage:perPageQuestionCount];

/**
 收到指定页码的问答数据
 #param history 问答数据，可能为空列表
 #param currentPage 当前页码，从0开始计数
 #param totalPage 最大页码，从0开始计数
 */
[self bjl_observe:BJLMakeMethod(self.room.roomVM, didLoadQuestionHistory:currentPage:totalPage:)
        observer:^BOOL(NSArray<BJLQuestion *> *history, NSInteger currentPage, NSInteger totalPage) {
    bjl_strongify(self);
    [self.tableView reloadData];
    return YES;
}];
```

- 创建、发布、取消发布、回复问答。

```
/**
 创建问答
 #param question 问题内容
 */
- (BJLError *)sendQuestion:(NSString *)question;
```

```
/**
 创建问答成功
 #param question 问答，包括问答 ID
 */
- (BJLObservable)didSendQuestion:(BJLQuestion *)question;

/**
 发布问答，需要先成功创建问答
 #param questionID 问答 ID
 */
- (BJLError *)publishQuestionWithQuestionID:(NSString *)questionID;

/**
 取消发布问答
 #param questionID 问答 ID
 */
- (BJLError *)unpublishQuestionWithQuestionID:(NSString *)questionID;

/**
 回复问答
 #param questionID 问答 ID
 #param state 该ID的问答的state
 #param reply 回复内容
 */
- (nullable BJLError *)replyQuestionWithID:(NSString *)questionID state:(BJLQuestionState)state reply:(NSString *)reply;
```

- 改变用户问答状态。

```
/**
 禁止提出问答的 userNumber 列表
```

```
 #discussion 列表包含禁止提出问答的 userNumber
 */
@property (nonatomic, readonly, nullable) NSSet<NSString *> *forbidQuestionList;

/**
 改变问答状态
 #param user user
 #param forbid 是否禁止问答
 */
- (BJLError *)switchQuestionForbidForUser:(BJLUser *)user forbid:(BOOL)forbid;
```

13. 网页

网页是用于老师发布给学生打开一个指定 URL 地址的网页，可以用于各种场景。

```
/**
 更新网页信息
 #param urlString 网址
 #param open YES：打开，NO：关闭
 #return BJLError
 */
[self.room.roomVM updateWebPageWithURLString:urlString open:publish];

/**
 收到网页信息更新
 #param urlString 网址
 #param open YES：打开，NO：关闭
 #param isCache 是否是缓存
 */
```

```
[self bjl_observe:BJLMakeMethod(self.room.roomVM, didUpdateWebPageWithURLString:open:isCache:)
        observer:(BJLMethodObserver)^BOOL(NSString *urlString, BOOL open, BOOL isCache) {
    bjl_strongify(self);
    if (open) {
        // open
    }
    else {
        // close
    }
    return YES;
}];
```

14. 学情报告

学情报告是用户在教室上课期间的统计数据报告，可以在下课后通过接口获取，使用 Webview 加载。

- 请求生成学情报告。

```
/**
 课后学情报告
 #return BJLStudyReportDataSource对象，每次调用返回不同对象
 */
- (BJLStudyReportDataSource *)getStudyReportDataSource;

/**
 课后学情报告，获取到返回的`NSURLRequest`结果后用 webview 打开
 #param userNumber 用户 number
```

```
#return NSURLRequest
 */
- (nullable NSURLRequest *)expressReportRequestWithUserNumber:(NSString *)userNumber;
```

15. 计时器

计时器是老师或者助教发布的计时工具，支持正计时器和倒计时器，发布后所有用户可以收到。

- 发布计时器。

```
/**
 发布计时器
 #param totalTime 计时总时长，单位 秒
 #param countDownTime 当前计时剩余计时时长，单位 秒
 #param isDecrease 是否为倒计时，NO表示正计时
 #return BJLError
 */
BJLError *error = [self.room.roomVM requestPublishTimerWithTotalTime:self.totalTime countDownTime:leftCountDownTime isDecrease:self.isDecrease];
```

- 收到计时器。

```
/**
 收到计时器信息更新
 #param time 倒计时时间
 #param open YES：发布，NO：关闭
 */
```

```
[self bjl_observe:BJLMakeMethod(self.room.roomVM, didReceiveTimerWithTotalTime:countDownTime:isD
        observer:(BJLMethodObserver)^BOOL (NSInteger totalTime, NSInteger countDownTime, BOOL isDecrease){
        bjl_strongify(self);
        return YES;
}];
```

- 暂停、取消计时器。

```
/**
 暂停计时器
 #param totalTime 计时总时长，单位 秒
 #param countDownTime 当前计时剩余计时时长，单位 秒
 #param isDecrease 是否为倒计时，NO表示正计时
 #return BJLError
 */
- (nullable BJLError *)requestPauseTimerWithTotalTime:(NSInteger)totalTime
                                        leftCountDownTime:(NSInteger)countDownTime
                                                isDecrease:(BOOL)isDecrease;
// 取消计时器
BJLError *error = [self.room.roomVM requestStopTimer];
```

16. 抢答器

抢答器是老师或者助教发布，优先抢到的学生回答。

- 发布抢答器。

```
/**
 请求发布抢答器
 #param time 抢答器倒计时
 #return BJLError
 */
BJLError *error = [self.room.roomVM requestPublishQuestionResponderWithTime:time];

// 收到抢答器开始信息
- (BJLObservable)didReceiveQuestionResponderWithT(NSInteger)time;
```

- 学生抢答。

```
BJLError *error = [self.room.roomVM submitQuestionResponder];
```

- 结束、撤销抢答器。

```
// 撤销抢答器
BJLError *error = [self.room.roomVM requestRevokeQuestionResponder];
// 关闭抢答器窗口
BJLError *error = [self.room.roomVM requestCloseQuestionResponder];
```

17. 答题器

答题由老师发布，学生接收并答题，不同于测验一般是课前准备，答题器适用于教室内临时发起的问题。每一个答题器都是 `BJLAnswerSheet` 对象，题目类型支持选择题或者判断题，

回答的选项是 `BJLAnswerSheetOption` 对象，可以定义每一个选项的名称以及是否是正确选项。答题器可以设置回答限制时间，在学生提交完成或者回答结束时，可以获取到参与人数、回答正确率等结果统计。

- 收到答题。

```
// 答题开始
[self bjl_observe:BJLMakeMethod(self.room.roomVM, didReceiveQuestionAnswerSheet:)
        observer:^BOOL(BJLAnswerSheet *answerSheet) {
    bjl_strongify(self);
    [self showProgressHUDWithText:@"答题开始"];
    [self showAnswerSheet];
    return YES;
}];
```

- 答题结束。

```
// 答题结束
[self bjl_observe:BJLMakeMethod(self.room.roomVM, didReceiveEndQuestionAnswerWithEndTime:)
        observer:(BJLMethodObserver)^BOOL(NSTimeInterval endTimeInterval) {
    bjl_strongify(self);
    [self showProgressHUDWithText:@"答题已结束"];
    [self clearAnswerSheet];
    return YES;
}];
```

- 提交答案。

```
/**
提交答案
 #param answerSheet 答题表：options 数组中的 BJLAnswerSheetOption 实例对应各个选项, 它的 seletced 属性表示该选项是否被选中
*/
 BJLError *error = [self.room.roomVM submitQuestionAnswer:answerSheet];
```

18. 抽奖

抽奖支持标准抽奖和口令抽奖二种抽奖方式。标准抽奖是在一系列用户中抽出中奖用户，口令抽奖是在老师发布指定口令，对在限定时间内发出口令的用户抽奖，SDK 支持参与和接收抽奖结果。

- 收到抽奖结果。

```
[self bjl_observe:BJLMakeMethod(self.room.roomVM, didReceiveLotteryResult:) observer:^BOOL(BJLLottery *lottery){
    bjl_strongify(self);
    if (BJLLotteryType_Command == lottery.type) {
    // 口令抽奖
    }
    else if (BJLLotteryType_Standard == lottery.type) {
        // 标准抽奖
    }
    return YES;
}];
```

- 收到口令抽奖开始。

```
[self bjl_observe:BJLMakeMethod(self.room.roomVM, didReceiveBeginCommandLottery:) observer:^BOOL{
    bjl_strongify(self);
    // 收到口令抽奖开始
    return YES;
}];
```

- 参与口令抽奖。

在聊天区发出了和要求口令相同的聊天内容后，调用该方法去参与口令抽奖。

```
[self.room.roomVM requestHitCommandLottery];

[self bjl_observe:BJLMakeMethod(self.room.roomVM, didReceiveHitCommandLottery:) observer:^BOOL{
    bjl_strongify(self);
    // 收到参与口令抽奖成功回调
    return YES;
}];
```

- 提交中奖信息。

中奖的用户可以填入联系方式，方便后期联系该用户。

```
/**
 提交中奖信息
 #param userName 用户名
```

```
 #param mobile 用户联系电话
 #param beginTime 抽奖开始时间
 #param completion 提交完成的回调
 */
- (void)submitLotteryUserName:(NSString *)userName
                       mobile:(NSString *)mobile
                    beginTime:(NSTimeInterval)beginTime
                   completion:(nullable void (^)(BOOL success))completion;
```

19. 转播

转播功能是在直播间转播另一个直播间的音视频、课件、画笔的功能，仅 WebRTC 大班课场景支持转播，可以用于多个直播间需要同时进行，但直播间存在部分共通的宣讲内容时使用。后台配置关联的转播教室，参考 `enableLiveBroadcast`，在教室内开启转播后，教室内的音视频用户数据将会切换成被转播教室的音视频数据，当前教室内无法采集音视频，课件和画笔也会同步被转播教室的内容，其他的功能保留当前教室的状态和使用方式。

```
/**
 开始转播
 #discussion 需要后台配置开启转播，关联了转播的教室，才能使用
 #discussion 转播仅转播目标教室的音视频、课件、画笔内容，同时将禁止这些功能，其他的功能如聊天等正常使用
 #discussion 大班课合流场景下使用
 #discussion 调用开始转播后，在未收到转播开始或者结束的回调之前的调用无效，但不会返回错误
 #return BJLError:
```

```
 BJLErrorCode_invalidUserRole  错误权限，要求老师或助教权限
 BJLErrorCode_invalidCalling  错误调用，当前教室不支持转播
 */
- (nullable BJLError *)startReceiveLiveBroadcast;

/** 停止转播，恢复教室转播前的状态 */
- (nullable BJLError *)stopReceiveLiveBroadcast;
```

20. 教室布局切换

大班课和专业小班课的标准 UI 存在不同的教室布局方式，对于大班课，有凸显主视频区域的布局，也能自由切换课件为主屏区域的布局，老师或者助教可以同步切换教室内的布局展示；对于专业小班课，默认是显示了黑板模块的板书布局，也有凸显用户音视频的画廊布局。

```
/**
 大班课 - 视频与 PPT 切换位置
 #param videoInMainPosition 视频是否切换到主位
 */
 - (nullable BJLError *)exchangeVideoPositonWithPPT:(BOOL)videoInMainPosition;
```

```
/**
 专业版小班课 - 更新教室布局请求
 #param roomLayout 窗口布局类型
 #return BJLError: 错误码参考 BJLErrorCode
 */
- (nullable BJLError *)updateRoomLayout:(BJLRoomLayout)roomLayout;
```

21. 积分系统

积分系统是基于教室内用户参与互动的情况，为每个用户的活跃度相关的数据化系统，可以在后台配置开启，支持设置参与互动可以获取到的积分数以及教室内能发放的总积分，SDK 提供获取到积分时的回调以及积分列表的情况。

```
/// 收到积分变动，只有老师收到该信令
/// @param remainBonus 教室剩余积分
/// @param success 积分变更是否成功，为false表示积分不够
- (BJLObservable)onReceiveBonusChange:(CGFloat)remainBonus success:(BOOL)success;

/// 收到积分增加信令，只有学生能收到该信令
/// @param bonus 本次新增积分
- (BJLObservable)onReceiveBonusIncreasing:(CGFloat)bonus;
```

```
/// 请求积分列表
/// @param type 排行榜类型
/// @param top top xx
/// @param userNumber 自己的userNumber
/// @param groupID 自己的分组id、id为0表示未分组或者是大班身份
- (nullable BJLError *)requestBonusRankListWithType:(BJLBonusListType)type top:(NSInteger)top userNumber:(NSString *)userNumber groupID:(NSInteger)groupID;
```