1.x > 2.x SDK 升级文档

SDK 精简

- 原 `BJVideoPlayer`、`BJPlayerManagerCore`、`BJPlaybackCore` SDK 合并为 `BJVideoPlayerCore`；
- 不再依赖 `LogStat`、`BJLLogStat`、`ASIHTTPRequest`、`Reachability`；
- `BJLiveUI`、`BJPlaybackUI` 和 `BJPlayerManagerUI` 使用 `BJLiveBase` 里的 `BJLAutolayout` 做自动布局，不再依赖 `Masonry`；

API 清理

- 原标记为 `DEPRECATED` 的类、属性、方法、枚举等全部删除，替代方案参考 1.x 版本 `DEPRECATED` 标记；

直播升级

- `BJLiveCore`、`BJLiveUI` 支持 WebRTC、支持同时观看多路视频流、支持专业小班课；
- `BJLiveCore` API 简化，`BJLRoom` 去掉 `vmsAvailable`、`inRoom` 属性，原来在 `vmsAvailable` 变为 `YES` 之后才能开始监听的属性和方法，现在创建 `BJLRoom` 实例后就可以开始监听，判断教室状态统一使用 `state` 属性。
- 2.0 及以上版本的 BJLiveCore SDK 支持同时播放一个用户的多路视频流，由 `BJLPlayingVM` 管理，对比旧版本 SDK，API 改动较大，集成方如果是从 1.x 版本或 `2.0.0-alpha`、`2.0.0-beta` 升级上来，且没有多视频流的需求，可以使用 BJLPlayingAdapterVM 管理音视频播放，此适配层支持旧版的单路流及主辅摄像头双路流模板，API 相对改动较小，具体使用方式可参考 旧版本 SDK 音视频模板适配 部分。
- 删除 `BJLSlideshowVM`、`BJLSlideVM`，课件由 `BJLDocumentVM` 管理。
- 新增 `BJLDrawingVM`，用于管理画笔。
- 移除部分 API，参考 直播文档相关部分。

点播、回放升级

- 由于原点播、回放、下载 Core SDK 的设计和实现存在缺陷，2.x 版本做了全面升级，新的 API 更简单、合理、易用；

下载升级

- 由于原下载设计和实现存在缺陷，2.x 版本做了全面升级，新的 API 更简单、合理、易用；
- 2.x 的下载通过 NSURLSession 实现，支持后台下载 —— App 被杀死之后在电量充足、接入 Wi-Fi 的情况下仍可继续完成下载；
- 由于 NSURLSession 限制，不建议使用下载队列 —— “等待中”的任务实际上是被暂停、后台下载会失效，可以设置 `BJLDownloadItem` 的 `priority` 来调整下载优先级 —— 但无法做到严格、精准的顺序控制；
- 由于 NSURLSession 限制，无法在下载过程中更改是否支持 4G 等设置，需要
  - 在 `application:will/didFinishLaunchingWithOptions:` 中 第一时间 设置 `BJLDownloadManager` 的 `delegate` 属性；
  - 实现 `BJLDownloadManagerDelegate` 协议中的 `downloadManager:configureNSURLSession:` 方法，在该方法中更改 `NSURLSessionConfiguration` 属性；

Autolayout 升级

- `Masonry` 从 iOS 11 发布之后到现在没再更新，`safeAreaLayoutGuide` 导致崩溃的问题迟迟无法解决，处于停止维护的状态，因此我们开发了 `BJLAutolayout` 来替代 `Masonry`，同时也减少了对外部代码的依赖；
- `BJLAutolayout` 的用法与 `Masonry` 非常相似，95% API 是兼容的，并且比 `Masonry` 更为安全、强大；

- 全局查找、替换即可完成大部分升级；
  - 全局查找 `bjlmas_` 并替换为 `bjl_`；
  - 全局查找 `mas_` 并替换为 `bjl_`；
  - 全局查找 `MASConstraint` 并替换为 `BJLConstraintAttributesMaker`、`BJLConstraintRelationMaker` 或 `BJLConstraint`；
  - 全局查找 `MASViewAttribute` 并替换为 `BJLConstraintTarget`；
  - 全局查找 `bjl_equalTo` 并替换为 `equalTo`（需要将简单类型转换为对象），或者 `equal.offset`、`equal.sizeOffset` 等；
  - 全局查找 `bjl_safeAreaLayoutGuideTop/Bottom/Left/Rigjt` 并替换为 `bjl_safeAreaLayoutGuide.bjl_top/bottom/left/right`；
  - 给 `BJLConstraint` 类型变量赋值时需要在 `make.xxxx.equal.to(xxxx)` 之后加 `.constraint`；
  - 不支持 `MASAttachKeys`；

KVO

- BJLiveBase 的 KVO 方法中 block 的参数调整；
- 1.x 中 block 接收 2 个参数 `old`、`now`；

```
typedef BOOL (^BJLPropertyFilter)(id _Nullable old, id _Nullable now);
typedef BOOL (^BJLPropertyObserver)(id _Nullable old, id _Nullable now);
typedef void (^BJLPropertiesObserver)(id _Nullable old, id _Nullable now);
```

- 2.x 中 block 接收 3 个参数，`value` —— 对应 1.x 的 now、`oldValue` —— 对应 1.x 的 old、`change` —— 新增参数；

```
typedef BOOL (^BJLPropertyFilter)(id _Nullable value, id _Nullable oldValue, BJLPropertyChange * _Nu
typedef BOOL (^BJLPropertyObserver)(id _Nullable value, id _Nullable oldValue, BJLPropertyChange *
typedef void (^BJLPropertiesObserver)(id _Nullable value, id _Nullable oldValue, BJLPropertyChange *
```

- 通过 `change` 可以获取更多 KVO 变化相关的信息；

```
@interface BJLPropertyChange : NSObject
@property (nonatomic, readonly) NSKeyValueChange kind; // NSKeyValueChangeKindKey
@property (nonatomic, readonly, nullable) id value, oldValue; // NSKeyValueChangeNewKey, NSKeyVa
@property (nonatomic, readonly, nullable) NSIndexSet *indexes; // NSKeyValueChangeIndexesKey
@property (nonatomic, readonly, getter=isPrior) BOOL prior; // NSKeyValueChangeNotificationIsPriorK
@property (nonatomic, readonly, nullable) void *context;
@property (nonatomic, readonly, getter=isChanged) BOOL changed; // value != oldValue
@property (nonatomic, readonly, getter=isDifferent) BOOL different; // value != oldValue && ![value
@end
```

- 兼容老代码

如果 KVO 代码较多，可以使用 `NSObject+BJLObserving.h` 中提供的兼容方法：

- 全局查找 `BJLProp` 并替换为 `BJLV1Prop`；
- 全局查找 `bjl_kvo` 并替换为 `bjlv1_kvo`，这样老代码依然可以运行；