微信小程序 回放SDK

概述

回放小程序SDK提供白板、聊天及loading三个组件供用户灵活组合界面，回放处理SDK是核心。

说明：由于微信会对小程序做不定期规则调整，为保证小程序业务正常，请及关注与更新小程序SDK版本。

SDK 下载地址

使用步骤

1、准备工作

SDK代码地址：http://git.baijiashilian.com/open-frontend/playback-weixin

1) playbackSDK文件夹放到项目根目录下

2) 在自己的页面中注册需要的回放组件

3) 小程序后台配置request合法域名

由于微信小程序SDK内部也需要通信，故需要用户配置request合法域名：

```
1. https://www.baijiayun.com (启用专属域名的客户，请填写为专属域名；)
2. (专属域名从百家云账号中心获取。例如： https://demo.at.baijiayun.com)
3. https://pro-www.baijiayun.com
4. https://click.baijiayun.com
5. https://img.baijiayun.com
```

4) 在页面onLoad后（或loading组件的loadingReady事件后）调用playback.init()

```
1.  playback.init({
2.    apiOrigin: '', // 与百家云约定的自定义域名，如'http://demo.at.baijiauyn.com', 若未约定则可以不传
3.      token: 'token值',
4.      class: {
5.      id: '教室ID',
6.      sessionId: 'sessionID',
7.      },
8.      // 用户信息(可不传，但会导致缺失用户听课记录)
9.      user: {
10.       number: '13147056',
11.       avatar: 'http://static.sunlands.com/newUserImagePath/13147056/40_40/13147056.jpg',
12.       name: 'xxx',
13.       type: 0
14.     }
15.   })
16.   .then(data => {
17.   // data为回传的媒体信息，用户自行将其赋值到相应的video或audio播放器
18.   });
```

2、组件使用

注意：微信开发工具模拟器与真机效果有差别，以真机为准。

loading组件

loading组件支持的绑定事件有： * loadingReady：在loading组件onLoad后触发 * loadEnded：在回放SDK加载完成时触发

WXML内容：

```
<!--demo/loading/loading.wxml-->

<!-- loadingReady在loading组件挂载到小程序页面上ready之后发出；loadEnded在回放数据加载完成时触发，此时loading组件会自动隐藏 -->
<loading logoSrc='{{logoSrc}}' styleInfo="{{loadingStyleInfo}}" bind:loadingReady="loadingReady" bind:loadEnded="loadEnded"></loading>
```

JS内容：

```
// demo/loading/loading.js
// 引入回放SDK
import playback from '../../playbackSDK/playback.js';

Page({

  /**
   * Page initial data
   */
  data: {
    // logSrc为进度条上方的logo图片，可以是小程序内的图片（必须是绝对路径，不然会找不到图片），也可以是在小程序后台注册过的网络地址
    logoSrc: '/demo/loading/m-loading.png',

    // 进度条样式
    loadingStyleInfo: {
      // 进度条前景色
      activeColor: '#1795ff',
      // 进度条背景色
      backgroundColor: '#aaa',
      // 进度条粗细
      strokeWidth: '5'
    },
  },

  /**
   * 必须等loading组件挂载到小程序上之后再触发回放SDK初始化，不然会有一些SDK发出的事件捕获不到
   */
  loadingReady: function () {
    // demo用例
    const params = {
      apiOrigin: '', // 与百家云约定的自定义域名，如'http://demo.at.baijiauyn.com', 若未约定则可以不传
        token: 'token值',
        class: {
        id: '教室ID',
        sessionId: 'sessionID',
        },
        // 用户信息(可不传，但会导致缺失用户听课记录)
        user: {
          number: '13147056',
          avatar: 'http://static.sunlands.com/newUserImagePath/13147056/40_40/13147056.jpg',
          name: 'xxx',
          type: 0
        }
    };
    playback.init(params).then(data => console.log(data));
  },
});
```

白板组件

WXML内容：

```
<!-- 引入的白板组件 -->
<whiteboard size='{{whiteboardSize}}' styleInfo="{{whiteboardStyleInfo}}"></whiteboard>
<!---->
```

白板有`size`和`styleInfo`两个属性，分别表示白板的宽高和样式。

```
// 单位为px
whiteboardSize: {
width: {number},
height: {number},
}

whiteboardStyleInfo: {
  backgroundColor: 'white', // 背景色
}
```

消息组件

消息组件支持的绑定事件有： * addMessage：在有新消息展示的时候触发 * imagePreview：在点击图片消息预览的时候触发

WXML内容：

```
<!-- 引入的消息组件 -->
<messageList styleInfo='{{messageListStyleInfo}}'></messageList>
<!---->
```

消息组件有`styleInfo`属性，表示消息组件的样式

```
// 消息列表组件相关配置
messageListStyleInfo: {
  //消息背景色
  messageBackground: '#f2f3f5',
  // 消息文本颜色
  contentColor: '#333'
}
```

一个完整的回放用例

小程序单个页面组成分为：`index.js` `index.json` `index.wxml` `index.wxss`

`index.json`中内容如下：

```
{
  "usingComponents": {
    "loading": "/playbackSDK/component/loading/loading",
    "whiteboard": "/playbackSDK/component/whiteboard/whiteboard",
    "messageList": "/playbackSDK/component/messageList/messageList"
  },
  "disableScroll": true
}
```

分别引入loading页、白板页、消息列表页

`index.wxml`内容如下。其中，为了用户体验，我们额外加了断网监听、视频速度及分辨率调整功能。这三个功能在demo文件内，不属于回放SDK

```
<!--demo/playback/playback.wxml-->
<view class="bjy-playback-view {{isWhiteboardFullScreen ? 'whiteboard-fullscreen-view' : ''}}">
  <!-- 断网监听（额外功能） -->
  <cover-view wx:if="{{!isConnected}}" class="offline-view">
```

```
<cover-view wx:if="{{!isConnected}}" class="offline-view">
  断网了哦~
</cover-view>
<!---->

<!-- 原生视频播放组件（具体使用见微信小程序官方文档 https://developers.weixin.qq.com/miniprogram/dev/component/video.html） -->
<video id='mainVideo' class="{{isLoading ? 'video-minify': ''}}" style='height: {{videoHeight}}px' controls custom-cache='{{false}}' src='{{currentVideoURL}}' bindplay='onVideoPlay' bindpause='onVideoPause' bindtimeupdate='onVideoTimeUpdate' bindended='onVideoEnded'
  bindwaiting='onVideoWaiting' binderror='onVideoError' bindfullscreenchange='onFullScreenChange'>
</video>
<!---->

<!-- 播放速度和分辨率调整区（额外功能，不需要可删除） -->
<view wx:if='{{!isWhiteboardFullScreen}}' class='video-controller'>
  <view class='rate-controller'>
    <picker bindchange="changeRate" value="{{rateArrIndex}}" range="{{rateArr}}">
      <view class="picker">
        播放速度：{{rateArr[rateArrIndex]}}x
      </view>
    </picker>
  </view>
  <view class='resolution-controller'>
    <picker bindchange="changeResolution" value="{{resolutionIndex}}" range="{{resolutionTextArr}}">
      <view class="picker">
        分辨率：{{currentResolutionText}}
      </view>
    </picker>
  </view>
</view>
<!---->

<!-- 白板及聊天区 -->
<view class="tab-container">

  <view wx:if='{{!isWhiteboardFullScreen}}' class='tab-header'>
    <text class="tab-btn {{tabIndex === 0 ? 'tab-btn-active' : ''}}" data-index='0' bindtap='changeTab'>聊天</text>
    <text class="tab-btn {{tabIndex === 1 ? 'tab-btn-active ' : ''}}" data-index='1' bindtap='changeTab'>白板</text>
  </view>

  <view class="tab-content {{tabIndex === 1 ? 'tab-content-whiteboard' : 'tab-content-message'}}">

    <!-- 引入的聊天组件 -->
    <messageList class='message-list-container' styleInfo='{{messageListStyleInfo}}'></messageList>
    <!---->

    <!-- 引入的白板组件 -->
    <whiteboard id="whiteboard" class="whiteboard" size='{{whiteboardSize}}' styleInfo="{{whiteboardStyleInfo}}"></whiteboard>
    <!---->
```

```
50.
51.     <!-- 控制白板全屏的按钮（额外功能） -->
52.     <cover-view wx:if='{{tabIndex === 1 && !isVideoFullScreen}}' class='fullscreen-btn'
    bindtap='toggleWhiteboardFullScreen'>
53.       <cover-image wx:if='{{isWhiteboardFullScreen}}' class="img" src="./img/whiteboard-shrink.png" />
54.       <cover-image wx:else class="img" src="./img/whiteboard-expand.png" />
55.     </cover-view>
56.     <!---->
57.   </view>
58.   <!---->
59.  </view>
60.
61.  <!-- 引入的loading组件 -->
62.  <loading wx:if="{{isLoading}}" class="loading-wrapper" styleInfo="{{loadingStyleInfo}}"
    bind:loadingReady="loadingReady" bind:loadEnded="roomLoadEnded"></loading>
63.  <!---->
64. </view>
```

index.wxss 文件内容：

```
1.  /* demo/playback/playback.wxss */
2.
3.  .bjy-playback-view {
4.    position: relative;
5.    display: flex;
6.    flex-direction: column;
7.    width: 100%;
8.    height: 100%;
9.    overflow: hidden;
10. }
11.
12. .loading-wrapper {
13.   position: absolute;
14.   top: 0;
15.   left: 0;
16.   right: 0;
17.   bottom: 0;
18.   width: 100%;
19.   height: 100%;
20.   z-index: 10;
21.   background-color: #fff;
22. }
23.
24. #mainVideo {
25.   width: 100%;
26. }
27.
28. .video-minify {
29.   width: 0;
30.   height: 0;
31. }
32.
33. .video-controller {
```

```
  display: flex;
  justify-content: space-evenly;
  align-items: center;
  font-size: 14px;
  line-height: 30px;
  background-color: #000;
  color: #fff;
}

.rate-controller, .resolution-controller {
  flex: 1;
  text-align: center;
}

.tab-container {
  flex: 1;
  display: flex;
  flex-direction: column;
  background-color: #fff;
}

.tab-header {
  display: flex;
}

.tab-btn {
  flex: 1;
  font-size: 14px;
  line-height: 29px;
  text-align: center;
  border-bottom: 1px solid #ccc;
}

.tab-btn-active {
  border-bottom: 1px solid #3ca2f1;
  color: #3ca2f1;
}

.tab-content {
  position: relative;
  flex: 1;
  transition: transform 0.5s;
}

.tab-content-message {
  transform: translateX(0);
}

.tab-content-whiteboard {
  transform: translateX(-100%);
}

```

```
/*
 * hack:
 * 白板canvas有bug，transform时不会立即销毁,此处直接将其移除屏幕
 */

.tab-content-message > .whiteboard {
  top: 200%;
}

/*
 * hack:
 * 白板canvas有bug，点击canvas会出现横向滚动，此处将聊天宽度设为0
 */

.tab-content-message > .whiteboard {
  width: 0;
}

.whiteboard {
  position: absolute;
  top: 0;
  left: 100%;
  width: 100%;
  height: 100%;
  display: flex;
  justify-content: center;
  align-items: center;
}

.message-list-container {
  position: absolute;
  top: 0;
  left: 0;
  width: 100%;
  height: 100%;
  overflow: hidden;
}

.fullscreen-btn {
  position: absolute;
  bottom: 20rpx;
  left: 200%;
  padding: 10rpx;
  width: 40rpx;
  height: 40rpx;
  z-index: 11;
  border-radius: 50%;
  background-color: rgba(0, 0, 0, 0.2);
  transform: translateX(-150%);
}

.whiteboard-fullscreen-view #mainVideo {
```

```
  width: 0;
  height: 0;
}

.whiteboard-fullscreen-view .tab-container {
  position: fixed;
  top: 0;
  left: 0;
  right: 0;
  bottom: 0;
  width: 100%;
  height: 100%;
}

.offline-view {
  position: fixed;
  top: 0;
  left: 0;
  right: 0;
  bottom: 0;
  z-index: 100;
  text-align: center;
  line-height: 100vh;
  overflow: hidden;
  background-color: rgba(0, 0, 0, 0.6);
  color: #fff;
}

```

`index.js` 文件内容：

```
// demo/playback/playback.js
// @author dujianhao

// 引入回放SDK
import playback from '../../playbackSDK/playback.js';

// 分辨率工具（可选）
import definitionUtil from './utils/definitionUtil';

// 网络监听工具（可选）
import networkUtil from './utils/networkUtil.js';

// 系统信息
const systemInfo = wx.getSystemInfoSync();

// 默认判断的seek最小距离
const SEEK_RANGE = 5;
let lastTimeUpdateTime = undefined;
let isFirstPlay = true;
let mediaContext = null;
let mediaWaitingTimer = null;

```

```
/**
 * 检测timeupdate是不是seek
 */
function checkIsSeek(currentTime) {

  let flag = false;
  // 小程序没有seek事件，通过timeupdate的差值判断是否seek
  if (lastTimeUpdateTime !== undefined && Math.abs(lastTimeUpdateTime - currentTime) > SEEK_RANGE)
{
    flag = true;
  }
  lastTimeUpdateTime = currentTime;
  return flag;
}


Page({
  /**
   * 页面的初始数据
   */
  data: {
    // 网络是否连接
    isConnected: true,

    videoSources: null,
    allVideos: null,
    videoWatermark: null,
    currentVideoSourceIndex: null,
    currentVideoSource: null,
    currentVideoURL: null,
    currentResolution: null,
    resolutionArr: [],
    resolutionIndex: 0,

    // 视频相关配置
    // 原生只支持0.5/0.8/1.0/1.25/1.5
    // https://developers.weixin.qq.com/miniprogram/dev/api/media/video/VideoContext.playbackRate.html
    rateArr: ['0.5', '0.8', '1.0', '1.25', '1.5'],
    rateArrIndex: 2,
    playRate: 1,
    isPlaying: false,
    isVideoFullScreen: false,
    videoHeight: undefined,

    // loading组件相关属性
    isLoading: true,
    loadingStyleInfo: {
      activeColor: '#1795ff',
      strokeWidth: '5',
      backgroundColor: '#aaa'
    },

```

```
    // 白板组件相关配置
    isWhiteboardFullScreen: false,
    showClear: false,
    whiteboardStyleInfo: {
      backgroundColor: 'white'
    },
    // 白板的宽高
    whiteboardSize: {
        width: systemInfo.windowWidth,
        height: Math.ceil(systemInfo.windowWidth * 3 / 4),
    },

    // 消息列表组件相关配置
    messageListStyleInfo: {
      messageBackground: '#f2f3f5',
      contentColor: '#333'
    },

    // tab项
    tabIndex: 0,
  },

  /**
   * 点击切换白板和消息列表tab
   * @param e
   */
  changeTab: function (e) {
    this.setData({
      tabIndex: +e.target.dataset.index
    });
  },

  /**
   * 等loading组件ready后开始playback数据获取
   * 不需要loading组件时将this.playbackInit()写到页面的onLoad函数末尾即可
   */
  loadingReady: function () {
    this.playbackInit();
  },

  /**
   * loading完成事件
   */
  roomLoadEnded: function () {
    this.setData({
      isLoading: false
    });
  },

  playbackInit: function() {
    playback.init({
        apiOrigin: '', // 与百家云约定的自定义域名，如'http://demo.at.baijiauyn.com', 若未约定则可以不传
```

```
      token: 'T4Ivx-DIWjgmvvoNNdcydzf3LcMMPzYEc4GNHnOv-QuVbTLv8dzJe_eqCFuUfaIw0nPLBgfsrQMKp0fXMnVKLQ',
      'class': {
        id: '18070683424562',
        sessionId: 201807200,
      },
      user: {
        number: '13147056',
        avatar: 'http://static.sunlands.com/newUserImagePath/13147056/40_40/13147056.jpg',
        name: 'xxx',
        type: 0
      }
    })
    .then(data => {
      if (!data) {
        return;
      }
      this.initData(data);
    });
  },


  /**
   * 获取回放数据之后初始化页面数据
   * @param data {Object}
   * data.title {String}: 回放标题
   * data.audio {String}: 回放音频
   * data.videoId {String}: 视频ID
   * data.videos {Object}: 所有视频
   * data.videoWatermark {String}: 视频水印，后台可配置
   * data.definition {Array}: 视频分辨率数组
   * data.defaultDefinition {String}: 默认分辨率
   */
  initData: function(data) {

    if (!(data && data.videos)) {
      return;
    }

    // 设置小程序标题
    wx.setNavigationBarTitle({
      title: data.title
    });

    // 包含所有分辨率，所有CDN的视频
    const allVideos = data.videos;

    const resolutionArr = definitionUtil.getDefinitionArr(Object.keys(data.videos));
    const resolutionTextArr = resolutionArr.map(item => {
      return definitionUtil.getDefinitionText(item);
    });

```

```
    // 后台可配置默认分辨率，此处读取后台默认配置
    const defaultDefinition = data.defaultDefinition;
    let currentResolution = resolutionArr[0];

    if (defaultDefinition && defaultDefinition !== currentResolution && allVideos[defaultDefinition]) {
      currentResolution = defaultDefinition;
    }

    const currentResolutionText = definitionUtil.getDefinitionText(currentResolution);

    const resolutionIndex = resolutionArr.indexOf(currentResolution);
    const videoSources = allVideos[currentResolution];
    let currentVideoSourceIndex = 0;
    const currentVideoSource = videoSources[currentVideoSourceIndex];

    this.setData({
      videoSources,
      currentVideoSourceIndex,
      currentVideoSource,
      currentVideoURL: currentVideoSource.url,
      currentResolution,
      resolutionArr,
      resolutionTextArr,
      currentResolutionText,
      resolutionIndex,
      allVideos,
      videoWatermark: data.videoWatermark,
    });

    // 发送媒体信息给回放SDK
    this.sendMediaInfo();
    // 获取video实例
    mediaContext = wx.createVideoContext('mainVideo', this);

    // 获取当前网络信息并toast提示
    networkUtil.getNetworkStatus();

    // 监听网络变化
    networkUtil.watchNetwork(
      () => {
        this.setData({
          isConnected: true
        });
      },
      () => {
        this.setData({
          isConnected: false
        });
      }
    );
    // 获取视频信息后更新界面布局
    this.updateLayout();
  }
```

```
231.
232.
233.  /**
234.   * 向回放SDK发送当前媒体信息
235.   * 首次播放的时候发送视频信息(可能存在切换cdn或清晰度，所以可能需要多次发送)
236.   */
237.  sendMediaInfo: function() {
238.    const sourceInfo = this.data.currentVideoSource;
239.    const mediaInfo = {};
240.    mediaInfo.cdn = sourceInfo.cdn;
241.    mediaInfo.filesize = sourceInfo.size;
242.    mediaInfo.totaltime = sourceInfo.duration;
243.    mediaInfo.resolution = this.data.currentResolution;
244.    mediaInfo.playfiletype = 'mp4';
245.    mediaInfo.contenttype = 0; // 播放内容的类型  0:正片 1:片头 2:片尾
246.
247.    playback.sendMediaInfo(mediaInfo);
248.  },
249.
250.  //*************************************************************************************/
251.  //****************************      视频相关监听      **********************************/
252.  //*************************************************************************************/
253.
254.  /**
255.   * 开始播放事件
256.   */
257.  onVideoPlay: function() {
258.    this.setData({
259.      isPlaying: true,
260.    });
261.    // 视频播放时告诉回放SDK开始播放了
262.    playback.play();
263.    isFirstPlay = false;
264.  },
265.
266.  /**
267.   * 暂停播放事件
268.   */
269.  onVideoPause: function() {
270.    this.setData({
271.      isPlaying: false,
272.    });
273.    // 视频暂停时告诉回放SDK暂停了
274.    playback.pause();
275.  },
276.
277.  /**
278.   * 时间更新事件
279.   */
280.  onVideoTimeUpdate: function(e) {
281.    // 视频时间更新则说明不再卡顿
282.    mediaWaitingTimer && (mediaWaitingTimer = clearTimeout(mediaWaitingTimer));
```

```
    if (isFirstPlay) {
      return;
    }
    const currentTime = e.detail.currentTime;
    const duration = e.detail.duration;

    // 小程序bug：卡顿或暂停后再播放会发出两次timeupdate事件，第二次为多余的事件，时间比第一次少2s左右
    const range = lastTimeUpdateTime - currentTime;
    if (range >= 0 && range < 3) {
      return;
    }

    // 微信video组件没有seek事件，此处必须自己判断
    if (checkIsSeek(currentTime)) {
      // 视频seek后需告诉回放SDKseek的位置
      playback.seek(currentTime);
    } else {
      playback.timeupdate(currentTime, duration);
    }
  },

  /**
   * 视频卡顿事件
   */
  onVideoWaiting: function() {
    console.log('waiting');
    wx.hideLoading();
    mediaWaitingTimer = clearTimeout(mediaWaitingTimer);
    if (this.data.isConnected) {
      wx.showToast({
        title: '有点卡哦~',
        icon: 'none',
        duration: 1000,
      });
      // 卡顿超过5秒则切CDN
      mediaWaitingTimer = setTimeout(() => {
        this.changeCDN();
      }, 5000);
    } else {
      wx.showToast({
        title: '断网了哦~',
        icon: 'none',
        duration: 1000,
      });
    }
  },

  /**
   * 视频出错事件
   */
  onVideoError: function() {
```

```
    console.log('error');
    // 播放出错且连着网，则切CDN
    if (this.data.isConnected) {
      this.changeCDN();
    }
  },

  /**
   * 结束播放事件
   */

  onVideoEnded: function(e) {
    // console.log('ended');
    // 告诉回放SDK视频播放结束
    playback.sendLog('endplay');
  },

  /**
   * 视频全屏触发
   */
  onFullScreenChange: function(e) {
    this.setData({
      isVideoFullScreen: e.detail.fullScreen
    });
  },

  //*****************************************************************************************/
  //*****************************************************************************************/
  //*****************************************************************************************/

  /**
   * 更改播放速率
   */
  changeRate: function(e) {
    const rateArrIndex = e.detail.value;
    const rateArr = this.data.rateArr;
    const playRate = +rateArr[rateArrIndex];
    this.setData({
      rateArrIndex,
      playRate,
    });
    // 小程序bug：改变速率的时候video会自己播放起来，而界面上还是显示的暂停
    // 此处多一次play，让界面显示正常播放样式
    mediaContext.play();
    wx.nextTick(() => {
      mediaContext.playbackRate(playRate);
    });
    playback.setSpeed(playRate);
  },

  /**
   * 更改CDN
   */
```

```
  changeCDN: function() {
    // 当前视频源index
    let currentVideoSourceIndex = this.data.currentVideoSourceIndex;
    // 当前视频源
    let currentVideoSource = this.data.currentVideoSource;

    // 视频源最后的index
    const lastSourcesIndex = currentVideoSource.length - 1;
    if (lastSourcesIndex === 0) {
      console.log('仅有唯一播放源，无法切换');
      return;
    }

    currentVideoSourceIndex = currentVideoSourceIndex >= lastSourcesIndex ? 0 :
++currentVideoSourceIndex;
    currentVideoSource = this.data.videoSources[currentVideoSourceIndex];

    this.setData({
      currentVideoSourceIndex,
      currentVideoSource,
    });
    this.changeMediaUrl(currentVideoSource.url, lastTimeUpdateTime);
  },

  /**
   * 更改视频URL
   */
  changeMediaUrl: function(newUrl, currentTime) {

    mediaContext.pause();

    // 小程序bug，切视频需要先将url置空，再赋值
    wx.nextTick(() => {
      this.setData({
        currentVideoURL: newUrl,
      });

      // 因video没有metadata事件，暴力解决切视频源后的继续播放问题
      const interval = setInterval(() => {
        if (this.data.isPlaying) {
          clearInterval(interval);
          return;
        }
        mediaContext.play();
        // 恢复到历史位置
        mediaContext.seek(currentTime);
      }, 500);
    });
    // 视频信息改变需要告诉回放SDK
    this.sendMediaInfo();
  },

```

```
  /**
   * 更改分辨率
   */
  changeResolution: function(e) {

    // 当前分辨率index
    const resolutionIndex = e.detail.value;

    // 当前分辨率
    const currentResolution = this.data.resolutionArr[resolutionIndex];

    // 当前分辨率文本
    const currentResolutionText = definitionUtil.getDefinitionText(currentResolution);

    // 当前分辨率下的视频源合集
    const videoSources = this.data.allVideos[currentResolution];
    // 重置视频源index为0
    const currentVideoSourceIndex = 0;
    // 当前视频源
    const currentVideoSource = videoSources[currentVideoSourceIndex];

    this.setData({
      videoSources,
      currentVideoSourceIndex,
      currentVideoSource,
      currentResolution,
      currentResolutionText,
      resolutionIndex,
    });

    this.changeMediaUrl(currentVideoSource.url, lastTimeUpdateTime);
  },

  /**
   * 白板全屏
   */
  toggleWhiteboardFullScreen: function() {
    const isWhiteboardFullScreen = this.data.isWhiteboardFullScreen;
    this.setData({
      isWhiteboardFullScreen: !isWhiteboardFullScreen
    });
    wx.nextTick(() => {
      this.updateLayout()
    });
  },

  /**
   * 通过视频尺寸来计算出剩余的白板尺寸
   * 视频宽度为屏幕宽
   * 视频下方的控制栏高30px
   * tab头高30px
   */
```

```
  updateLayout: function () {

    const CONTROLLER_HEIGHT = 30;
    const TAB_HEADER_HEIGHT = 30;

    const {
      windowWidth,
      windowHeight
    } = systemInfo;
    // 白板默认宽高
    let whiteboardSize = {
      width: windowWidth,
      height: Math.ceil(windowWidth * 3 / 4)
    };

    // 视频默认高度240px
    let videoHeight = 240;
    // 白板全屏时设置白板宽高为窗口宽高
    if (this.data.isWhiteboardFullScreen) {
      whiteboardSize = {
        width: windowWidth,
        height: windowHeight
      }
    } else {
      const currentVideoSource = this.data.currentVideoSource;
      const videoRatio = currentVideoSource.height / currentVideoSource.width;
      videoHeight = windowWidth * videoRatio;
      // 白板高度为 窗口高度 - 视频高度 - 控制条高度 - tab高度
      const whiteboardHeight = windowHeight - videoHeight - CONTROLLER_HEIGHT -
TAB_HEADER_HEIGHT;

      whiteboardSize = {
        width: windowWidth,
        height: whiteboardHeight
      }
    }
    this.setData({
      videoHeight,
      whiteboardSize,
    });
  },

  /**
   * 生命周期函数--监听页面加载
   */
  onLoad: function(options) {
    // 设置屏幕不锁屏
    wx.setKeepScreenOn({
      keepScreenOn: true
    });
  },
});
```

网络监听工具networkUtil.js内容：

```
function watchNetwork(successCallback, errorCallback) {

  // 监听网络变化
  wx.onNetworkStatusChange(res => {
    if (res.isConnected) {
      wx.showToast({
        title: `网络连接改变，当前网络:${res.networkType.toUpperCase()}`,
        icon: 'none'
      });
      successCallback(res);
    } else {
      wx.showToast({
        title: `网络连接出错`,
        icon: 'none'
      });
      errorCallback(res);
    }
  })
}

function getNetworkStatus () {
  return new Promise((resolve, reject) => {
    wx.getNetworkType({
      complete: res => {
        if (res.errMsg === 'getNetworkType:ok') {
          wx.showToast({
            title: `当前网络:${res.networkType.toUpperCase()}`,
            icon: 'none'
          });
        } else {
          wx.showToast({
            title: `网络连接出错`,
            icon: 'none'
          });
        }
        resolve(res);
      }
    });
  })
}

export default {
  watchNetwork,
  getNetworkStatus,
}
```

分辨率工具`definitionUtil.js`内容：

```
/*
 * @file 格式化清晰度
 * @author dujianhao
 */

const videoConfig = {

  VIDEO_DEFINITION_STANDARD: 'low',

  VIDEO_DEFINITION_HIGH: 'high',

  VIDEO_DEFINITION_SUPER: 'super',

  VIDEO_DEFINITION_SUPERHD: 'superHD',

  VIDEO_DEFINITION_720: '720p',

  VIDEO_DEFINITION_1080: '1080p',
};

const definitionArr = [
    videoConfig.VIDEO_DEFINITION_STANDARD,
    videoConfig.VIDEO_DEFINITION_HIGH,
    videoConfig.VIDEO_DEFINITION_SUPER,
    videoConfig.VIDEO_DEFINITION_SUPERHD,
    videoConfig.VIDEO_DEFINITION_720,
    videoConfig.VIDEO_DEFINITION_1080,
];
function getDefinitionArr (arr) {
  return definitionArr.filter(value => {
    return ~arr.indexOf(value);
  });
}

const definitionTextArr = {
  [videoConfig.VIDEO_DEFINITION_STANDARD]: '标清',
  [videoConfig.VIDEO_DEFINITION_HIGH]: '高清',
  [videoConfig.VIDEO_DEFINITION_SUPER]: '超清',
  [videoConfig.VIDEO_DEFINITION_SUPERHD]: '超清HD',
  [videoConfig.VIDEO_DEFINITION_720]: '720p',
  [videoConfig.VIDEO_DEFINITION_1080]: '1080p',
};

function getDefinitionText (definition) {
  return definitionTextArr[definition];
}

export default {
  getDefinitionArr,
  getDefinitionText,
}
```

webview 嵌套百家云回放页面

- 提供合法域名校验文件放置于百家云服务器
- 复制百家云后台回放链接如

```
1. https://b62335648.at.baijiayun.com/web/playback/index?classid=19112058506594&session_id=201911200&token=Q5SOKStj0uHRK0w_qDAQFz9EExYsfrd2X-fUuCaOqrwiLLhQNWecHg9y4aYdBkZYKRmmy3XBDnEKp0fXMnVKLQ
```

- 链接后拼接上以下参数

```
1. &dsp=1&disable_ppt_animate=1
```

- 小程序中赋值webview src

```
1. <web-view src="https://b62335648.at.baijiayun.com/web/playback/index?classid=19112058506594&session_id=201911200&token=Q5SOKStj0uHRK0w_qDAQFz9EExYsfrd2X-fUuCaOqrwiLLhQNWecHg9y4aYdBkZYKRmmy3XBDnEKp0fXMnVKLQ&dsp=1&disable_ppt_animate=1"></web-view>
```