

入门示例

此处提供一个基于观看页 SDK 模块结合使用的入门示例，包含以下示例内容

- [播放器模块-入门示例](#)
- [聊天室模块-入门示例](#)

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta
      name="viewport"
      content="width=device-width, initial-scale=1.0"
    />
    <title>Demo</title>
    <style>
      html,
      body {
        padding: 0;
        margin: 0;
      }

      .demo-player-container {
        width: 640px;
        height: 360px;
        padding: 16px;
        margin-bottom: 80px;
      }

      #player {
        width: 100%;
        height: 100%;
        position: relative;
      }

      .demo-player-control {
        display: flex;
        margin-top: 24px;
        gap: 12px;
      }

      .demo-chat-container {
        min-width: 640px;
        max-width: 80vw;
        height: 360px;
        padding: 16px;
      }

      .demo-chat__list {
        width: 100%;
        height: 100%;
        overflow-y: auto;
      }

      .demo-chat-control {
        display: flex;
        margin-top: 24px;
        gap: 12px;
      }
    </style>
  </head>
```

```

<body>
  <div class="demo-player-container">
    <div id="player"></div>
    <div class="demo-player-control">
      <button class="demo-player-control__play-btn">播放</button>
      <button class="demo-player-control__pause-btn">暂停</button>
    </div>
  </div>

  <div class="demo-chat-container">
    <div class="demo-chat-input-control">
      <input
        class="demo-chat-input-control__input"
        type="text"
        placeholder="请输入纯文本消息"
      />
      <button class="demo-chat-input-control__send-btn">发送普通文本消息</button>
      <button class="demo-chat-input-control__send-btn--custom">发送自定义消息</button>
      <button class="demo-chat-input-control__send-like">
        点赞：
        <span id="like-count">0</span>
      </button>
      <button class="demo-chat-input-control__load-prev-btn">加载上一页</button>
      <button class="demo-chat-input-control__load-next-btn">加载下一页</button>
    </div>
    <ul class="demo-chat__list"></ul>
  </div>

  <!-- 此处使用在线JS链接演示，开发者可使用npm包形式安装 -->
  <script src="https://websdk.videooc.net/live-watch-sdk/latest/lib/polyv-live-watch-sdk.umd.js"></script>
  <script>
    const {
      createWatchCore,

      PolyvWatchCoreEvents,
      ChannelEvents,
      PlayerEvents,
      ChatEvents,

      LiveStatus,
      ChatRequestHistoryMode,
      ChatMsgSource,
    } = window.PolyvLiveWatchSDK;

    /** 全局变量 */
    var globalStore = {
      watchCore: null, // 观看页核心实例
      playerInited: false, // 播放器是否初始化完成
      playStatus: false, // 播放状态
      liveStatus: LiveStatus.End, // 直播流状态
      chatConnected: false, // 是否连接聊天室
      chatHistoryData: [], // 聊天历史数据
    };

    /**
     * 绑定播放器 UI 事件
  
```

```

*/
function bindPlayerUIEvent(watchCore) {
  const $playBtn = document.querySelector('.demo-player-control__play-btn');
  const $pauseBtn = document.querySelector('.demo-player-control__pause-btn');

  $playBtn.addEventListener('click', () => {
    if (!globalStore.playerInitied) return;

    watchCore.player.play();
  });

  $pauseBtn.addEventListener('click', () => {
    if (!globalStore.playerInitied) return;

    watchCore.player.pause();
  });
}

/**
 * 绑定聊天室 UI 事件
 */
function bindChatUIEvent(watchCore) {
  const $input = document.querySelector('.demo-chat-input-control__input');
  const $sendBtn = document.querySelector('.demo-chat-input-control__send-btn');
  const $customSendBtn = document.querySelector('.demo-chat-input-control__send-btn--custom');
  const $sendLike = document.querySelector('.demo-chat-input-control__send-like');
  const $loadPrevBtn = document.querySelector('.demo-chat-input-control__load-prev-btn');
  const $loadNextBtn = document.querySelector('.demo-chat-input-control__load-next-btn');

  $sendBtn.addEventListener('click', () => {
    if (!globalStore.chatConnected) return;

    const content = $input.value.trim();
    if (!content) return;

    $input.value = '';
    // 发送文本消息
    watchCore.chat.sendSpeakMsg({
      content,
    });
  });

  $customSendBtn.addEventListener('click', () => {
    if (!globalStore.chatConnected) return;

    const content = $input.value.trim();
    if (!content) return;

    $input.value = '';
    // 发送自定义消息
    watchCore.chat.sendCustomMessage({
      data: { content },
    });
  });

  $sendLike.addEventListener('click', () => {

```

```

    if (!globalStore.chatConnected) return;

    // 发送点赞消息
    // 注意：此处仅作参考，真实场景请使用节流机制，避免高并发
    watchCore.chat.sendLike(1);
  });

  // 加载上一页数据
  $loadPrevBtn.addEventListener('click', loadPrevChatHistory);

  // 加载下一页数据
  $loadNextBtn.addEventListener('click', loadNextChatHistory);
}

/**
 * 绑定 UI 事件
 */
function bindUIEvent(watchCore) {
  bindPlayerUIEvent(watchCore);
  bindChatUIEvent(watchCore);
}

/**
 * 绑定 watch-sdk 事件
 */
function bindWatchSdkEvent(watchCore) {
  // 监听核心 - setup 钩子
  watchCore.eventEmitter.on(PolyvWatchCoreEvents.WatchCoreSetup, () => {
    console.info('观看页核心实例安装完成');
  });

  // 监听核心 - 连接聊天室
  watchCore.eventEmitter.on(PolyvWatchCoreEvents.WatchCoreConnected, () => {
    console.info('观看页核心实例连接聊天室成功');
    globalStore.chatConnected = true;
  });

  // 监听直播流变化
  watchCore.channel.eventEmitter.on(ChannelEvents.LiveStatusChange, ({ liveStatus }) => {
    if (globalStore.liveStatus === liveStatus) return;

    globalStore.liveStatus = liveStatus;
    console.warn('流状态变化', liveStatus);

    // 直播流变化时重新创建播放器
    if (globalStore.playerInited) {
      globalStore.playerInited = false;
      createPlayer(watchCore);
    }
  });

  // 监听播放器-初始化事件
  watchCore.player.eventEmitter.on(PlayerEvents.PlayerInited, () => {
    const playerInfo = watchCore.player.getPlayerInfo();
    globalStore.playerInited = playerInfo.playerInited;
  });
}

```

```

        console.warn('播放器初始化完成');
        console.log('当前是否使用无延迟播放器: ', playerInfo.isLowLatency);
    });

    // 监听播放器-播放信息变化
    watchCore.player.eventEmitter.on(PlayerEvents.PlayerInfoChange, ({ playerInfo }) => {
        globalStore.playStatus = playerInfo.playStatus;
    });

    // 监听播放器-播放时间改变
    watchCore.player.eventEmitter.on(
        PlayerEvents.TimeUpdate,
        ({ currentTime, durationTime }) => {
            console.warn('播放器时间变化: ', { currentTime, durationTime });
        },
    );

    // 监听播放器-播放事件
    watchCore.player.eventEmitter.on(PlayerEvents.PlayerPlaying, () => {
        console.warn('播放回调, 播放视频中');
    });

    // 监听播放器-暂停事件
    watchCore.player.eventEmitter.on(PlayerEvents.PlayerPause, () => {
        console.warn('暂停回调, 已暂停视频播放');
    });

    // 监听聊天室-超出直播间最大在线人数
    watchCore.chat.eventEmitter.on(ChatEvents.OverMaxOnlineCount, () => {
        alert('已超出直播间最大在线人数 ! ');
    });

    // 监听聊天室-点赞数回调
    watchCore.chat.eventEmitter.on(ChatEvents.ChatLikeCountChange, data => {
        console.log('实时点赞数: ', data.realtimeLikes);
        $likeCount = document.querySelector('#like-count');
        $likeCount.innerText = data.realtimeLikes;
    });

    // 监听聊天室-聊天消息事件
    watchCore.chat.eventEmitter.on(ChatEvents.ChatMessage, ({ chatMsg }) => {
        if (
            chatMsg.msgSource === ChatMsgSource.CustomerMessage ||
            chatMsg.msgSource === ChatMsgSource.CustomMessage
        ) {
            console.warn('自定义聊天消息回调: ', chatMsg);
        } else {
            console.warn('普通聊天消息回调: ', chatMsg);
        }

        renderMsgList([chatMsg]);
    });
}

/**
 * 绑定事件

```

```

*/
function bindEvent(watchCore) {
  bindUIEvent(watchCore);
  bindWatchSdkEvent(watchCore);
}

/**
 * 安装核心实例和授权
 * 注意：本示例使用无条件观看，故首次登录的用户信息会缓存有效期，如希望用户信息不缓存，请使用授权验证或者使用频道观看
限制（如独立授权），二选一即可，如果使用授权验证，下面 频道观看限制 的判断代码就可以去掉，改成授权验证流程，建议带上await，
然后再次执行await watchCore.setup()，重新安装watchCore
*/
async function setupWatchCore(watchCore) {
  await watchCore.setup();

  // 授权验证: https://help.polyv.net/index.html#/live/js/new_sdk/live_watch_sdk/articles/verify-
next/overview

  // 频道观看限制
  if (!watchCore.auth.isAuthorized()) {
    // 观众点击观看页入口或执行观看条件授权，如：无条件授权
    const result = await watchCore.auth.verifyNoneAuth();
    if (!result.success) {
      console.error('授权失败了!!', result.failReason);
      return;
    }

    // 当观众执行观看条件授权成功，需要重新安装 watchCore
    await watchCore.setup();
  }
}

/**
 * 创建播放器
*/
async function createPlayer(watchCore) {
  // 获取回放选项，此次仅以单个回放为例
  function __getPlayerPlaybackOptions() {
    const liveStatus = watchCore.channel.getLiveStatus();

    if (liveStatus !== LiveStatus.Playback) return;

    const playbackTarget = watchCore.playback.getSinglePlaybackTarget();
    watchCore.playback.setCurrentPlaybackTarget(playbackTarget);

    return playbackTarget.playbackOptions;
  }

  // 具体文档参数见：
https://help.polyv.net/index.html#/live/miniprogram/live\_watch\_miniprogram\_sdk/articles/modules/player/player?id=classmethoddocusplayermodule\_setupplayer
  await watchCore.player.setupPlayer({
    container: '#player',
    lowLatency: false, // 是否无延迟播放
    showController: false, // 是否显示播放器控制栏
    playbackOptions: __getPlayerPlaybackOptions(), // 回放参数
  });
}

```

```

    });
  }

  /**
   * 渲染聊天列表
   * @warn 此处是消息渲染简单示例，真实场景请务必做好虚拟列表来处理高并发场景
   */
  function renderMsgList(data, reset = false) {
    const $msgList = document.querySelector('.demo-chat__list');
    const fragment = document.createDocumentFragment();
    data
      // 此处只处理了发言类型的消息
      .filter(chatMsg => chatMsg.msgSource === ChatMsgSource.Speak)
      .forEach(chatMsg => {
        const li = document.createElement('li');
        li.textContent = `${chatMsg.user.nick}: ${chatMsg.content || ''}`;
        fragment.appendChild(li);
      });

    if (reset) {
      $msgList.innerHTML = '';
    }

    $msgList.appendChild(fragment);
  }

  /**
   * 获取聊天消息所在的相同时间戳的数量
   * @param chatMsg 聊天消息
   * @param index 聊天消息所在历史数据列表的下标
   * @param mode 排序
   */
  function findSameTimestampMsgCount(chatMsg, index, mode) {
    const historyData = globalStore.chatHistoryData;
    let count = 1;
    const currentIndex = index;

    const checkWhileJudge = () => {
      const nextMsg =
        mode === ChatRequestHistoryMode.Prev
          ? historyData[currentIndex - 1]
          : historyData[currentIndex + 1];
      return nextMsg && chatMsg.time === nextMsg.time;
    };
    let isWhileJudge = checkWhileJudge();

    while (isWhileJudge) {
      count += 1;
      isWhileJudge = checkWhileJudge();
    }

    return count;
  }

  /**
   * 加载上一页历史聊天消息

```



```

*/
async function loadPrevChatHistory() {
  const historyLength = globalStore.chatHistoryData.length;
  const watchCore = globalStore.watchCore;
  if (historyLength === 0 || !watchCore) return;

  const firstMsg = globalStore.chatHistoryData[0];
  const count = findSameTimestampMsgCount(firstMsg, 0, ChatRequestHistoryMode.Prev);

  const result = await watchCore.chat.getChatHistoryByTime({
    timestamp: firstMsg.time,
    count,
    mode: ChatRequestHistoryMode.Prev,
  });

  console.warn('加载上一页历史聊天消息: ', result);

  globalStore.chatHistoryData = result.concat(globalStore.chatHistoryData);
  renderMsgList(globalStore.chatHistoryData, true);
}

/**
 * 加载下一页历史聊天消息
 */
async function loadNextChatHistory() {
  const historyLength = globalStore.chatHistoryData.length;
  const watchCore = globalStore.watchCore;
  if (historyLength === 0 || !watchCore) return;

  const lastMsg = globalStore.chatHistoryData[historyLength - 1];
  const count = findSameTimestampMsgCount(
    lastMsg,
    historyLength - 1,
    ChatRequestHistoryMode.Next,
  );

  const result = await watchCore.chat.getChatHistoryByTime({
    timestamp: lastMsg.time,
    count,
    mode: ChatRequestHistoryMode.Next,
  });

  console.warn('加载下一页历史聊天消息: ', result);

  globalStore.chatHistoryData = globalStore.chatHistoryData.concat(result);
  renderMsgList(result);
}

/**
 * 创建聊天列表
 */
async function createChatList(watchCore) {
  // 此处只获取常用的聊天历史数据
  const historyData = await watchCore.chat.getChatHistoryByTime({
    mode: ChatRequestHistoryMode.Prev,
  });

```

```
    globalStore.chatHistoryData = historyData;

    renderMsgList(historyData);
}

/**
 * 入口函数
 */
async function main() {
    // 创建核心, 文档见:
https://help.polyv.net/index.html#/live/miniprogram/live\_watch\_miniprogram\_sdk/articles/core/sdk-core
    const watchCore = createWatchCore({
        channelId: 'xxx', // 请使用一个"公开观看"的频道
        userInfo: {
            // 用户信息
            userId: 'polyv-123456',
            nick: 'polyv-测试昵称',
            actor: 'polyv-测试头衔',
        },
    });

    globalStore.watchCore = watchCore;

    // 绑定相关事件
    bindEvent(watchCore);
    // 安装核心
    await setupWatchCore(watchCore);

    // 连接聊天室
    watchCore.connect();

    // 创建播放器
    createPlayer(watchCore);
    // 创建聊天列表
    createChatList(watchCore);
}

main();
</script>
</body>
</html>
```