入门示例

此处提供一个连麦模块的基础使用示例，该示例包含以下内容

1. 连接聊天室，初始化连麦功能，支持本地设备调试
2. 支持观众申请连麦，取消申请连麦操作，讲师邀请观众上麦功能
3. 支持不使用内置 UI，自行实现结束连麦、切换摄像头开关、切换麦克风开关、切换镜像功能
4. 支持响应后台配置处理视频连麦、语音连麦逻辑
5. 部分和播放器模块配合使用的逻辑
6. 支持自定义邀请上麦 UI（acceptInvite / refuseInvite / getInviteCountDown）
7. 支持本地预览摄像头画面（startPreview / stopPreview / getPreviewVolume）
8. 支持预设本地音视频开关（getLocalMuteStatus / setLocalMuteStatus）

更多内容请结合文档参考 观看页 SDK 及其配套的开源项目

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

      .demo-connect-mic-container {
        padding: 12px 16px;
      }

      .demo-connect-mic-base-info {
        margin-bottom: 16px;
      }

      .demo-connect-mic-control {
        margin-bottom: 16px;
      }

      .demo-connect-mic-list {
        display: flex;
      }

      .demo-connect-mic-list-item {
        width: 200px;
        border: 1px solid #ccc;
        border-radius: 8px;
        margin-right: 12px;
      }

      .demo-connect-mic-list-item__content {
        height: 120px;
      }

      .demo-connect-mic-list-item__content video {
        transform: none !important;
      }

      .demo-connect-mic-list-item__content--no-mirror video {
        transform: scaleX(-1) !important;
      }

      .demo-connect-mic-list-item__info {
        display: flex;
        flex-direction: column;
        padding: 8px;
        line-height: 1.5;
        word-wrap: break-word;
      }
```

```
    .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;
    }
  </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>
  <!--
   1、一般需要讲师端开启连麦功能，观众端才能申请连麦，或者被讲师邀请上麦
   2、连麦功能需要需要连接保利威聊天室，在直播开始时才能正常使用
  -->
  <div class="demo-connect-mic-container">
    <div class="demo-connect-mic-base-info">
      <div>远程连麦状态： <span id="open-mic-status"></span></div>
      <div>当前用户连麦状态: <span id="connect-mic-status"></span></div>
      <div>邀请倒计时: <span id="invite-count-down"></span></div>
      <div>本地音视频开关: <span id="local-mute-status"></span></div>
    </div>

    <div class="demo-connect-mic-control">
      <button class="demo-connect-mic-control__open-setting-btn">
        设备检测
      </button>
      <button class="demo-connect-mic-control__apply-btn">申请连麦</button>
      <button class="demo-connect-mic-control__cancel-apply-btn">
        取消申请
      </button>
      <button class="demo-connect-mic-control__hang-up-btn">结束连麦</button>
      <button class="demo-connect-mic-control__video-toggle">
        切换摄像头开关
      </button>
      <button class="demo-connect-mic-control__audio-toggle">
        切换麦克风开关
      </button>
      <button class="demo-connect-mic-control__mirror-toggle">
```

```
          切换镜像
        </button>
        <button class="demo-connect-mic-control__accept-invite-btn">
          接受邀请上麦
        </button>
        <button class="demo-connect-mic-control__refuse-invite-btn">
          拒绝邀请上麦
        </button>
        <button class="demo-connect-mic-control__start-preview-btn">
          开启本地预览
        </button>
        <button class="demo-connect-mic-control__stop-preview-btn">
          停止本地预览
        </button>
        <button class="demo-connect-mic-control__toggle-local-mute-btn">
          切换本地音视频开关
        </button>
      </div>

      <div class="demo-connect-mic-list">
        <!-- <div class="demo-connect-mic-list-item"></div> -->
      </div>

      <div class="demo-connect-mic-preview" style="margin-top: 16px;">
        <div>本地预览：</div>
        <div id="preview-container" style="width: 200px; height: 150px; border: 1px solid #ccc; border-radius:
8px;"></div>
        <div>预览音量：<span id="preview-volume">0</span></div>
      </div>
    </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.videocc.net/live-watch-sdk/latest/lib/polyv-live-watch-sdk.umd.js"></script>
    <script>
      const {
        createWatchCore,

        PolyvWatchCoreEvents,
        ChannelEvents,
        PlayerEvents,
        ChatEvents,
        ConnectMicEvents,

        LiveStatus,
        ConnectMicType,
        ConnectMicStatus,
      } = 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
        connectMicInited: false, // 是否初始化连麦功能
```

```
  connectMicInfo: null, // 连麦配置信息
  networkInfo: null, // 连麦网络信息
  connectMicList: [], // 连麦列表
};

/**
 * 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.playerInited) return;

    watchCore.player.play();
  });

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

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

/**
 * 检查是否支持连麦
 */
function checkConnectMicEnabled() {
  if (globalStore.liveStatus !== LiveStatus.Live) {
    console.warn("请先发起直播！");
    return;
  }

  const enabled =
    globalStore.connectMicInited &&
    !!globalStore.connectMicInfo &&
    globalStore.connectMicInfo.supportConnectMic;

  if (!enabled) {
    console.warn(
      "当前频道配置或浏览器环境不支持连麦功能！请检查配置和使用的浏览器版本！"
    );
  }

  return enabled;
}

function checkApplyConnectMicEnabled() {
  if (!checkConnectMicEnabled()) {
    return false;
  }
```

```
  const { openMicStatus, inviteStatus } = globalStore.connectMicInfo;

  return openMicStatus || inviteStatus;
}

/**
 * 检查是否正在连麦
 */
function checkIsConnectMicing() {
  if (!checkConnectMicEnabled()) {
    return false;
  }

  const status = globalStore.connectMicInfo.connectMicStatus;
  return [
    ConnectMicStatus.Publishing,
    ConnectMicStatus.Connected,
  ].includes(status);
}

/**
 * 获取本地连麦流对象
 */
function getLocalMicItem() {
  const micList = globalStore.connectMicList;
  const localIndex = micList.findIndex((micItem) => micItem.isLocal);
  return localIndex === -1 ? undefined : micList[localIndex];
}

/**
 * 生成连麦区域元素 id
 */
const generateMicId = (micItem) =>
  micItem.isSelf ? "connect-mic-self" : "connect-mic-" + micItem.streamId;

/**
 * 生成连麦信息区域元素 id
 */
const generateMicInfoId = (micItem) =>
  micItem.isSelf
    ? "connect-mic-info-self"
    : "connect-mic-info" + micItem.streamId;

/**
 * 渲染连麦信息-摄像头状态
 */
function renderMicInfoVideoText({ isVideoMuted, micItem }) {
  const $micItemInfo = document.getElementById(
    generateMicInfoId(micItem)
  );
  const $micInfoVideoText = $micItemInfo.querySelector(
    ".demo-connect-mic-list-item__info__video"
  );

  $micInfoVideoText.innerText = `摄像头状态：${
```

```
    isVideoMuted ? "关闭" : "开启"
  }`;
}

/**
 * 渲染连麦信息-麦克风状态
 */
function renderMicInfoAudioText({ isAudioMuted, micItem }) {
  const $micItemInfo = document.getElementById(
    generateMicInfoId(micItem)
  );
  const $micInfoAudioText = $micItemInfo.querySelector(
    ".demo-connect-mic-list-item__info__audio"
  );

  $micInfoAudioText.innerText = `麦克风状态：${
    isAudioMuted ? "关闭" : "开启"
  }`;
}

/**
 * 渲染连麦信息-镜像状态
 */
function renderMicInfoMirrorText({ mirrorEnabled, micItem }) {
  const $micItemInfo = document.getElementById(
    generateMicInfoId(micItem)
  );
  const $micInfoMirrorText = $micItemInfo?.querySelector(
    ".demo-connect-mic-list-item__info__mirror"
  );

  if ($micInfoMirrorText) {
    $micInfoMirrorText.innerText = `镜像状态：${
      mirrorEnabled ? "开启" : "关闭"
    }`;
  }

  // 更新镜像CSS类
  const $micItem = document.getElementById(generateMicId(micItem));
  if ($micItem && micItem.isLocal) {
    if (mirrorEnabled) {
      $micItem.classList.remove(
        "demo-connect-mic-list-item__content--no-mirror"
      );
    } else {
      $micItem.classList.add(
        "demo-connect-mic-list-item__content--no-mirror"
      );
    }
  }
}

/**
 * 渲染连麦列表
 */
function renderConnectMicList() {
```

```
        const connectMicList = globalStore.connectMicList;

        const $connectMicList = document.querySelector(
          ".demo-connect-mic-list"
        );

        $connectMicList.innerHTML = connectMicList
          .map(
            (micItem) => `
            <div class="demo-connect-mic-list-item" >
              <div class="demo-connect-mic-list-item__content ${
                micItem.isLocal && !micItem.mirrorEnabled
                  ? "demo-connect-mic-list-item__content--no-mirror"
                  : ""
              }" id="${generateMicId(micItem)}"></div>
              <div class="demo-connect-mic-list-item__info" id="${generateMicInfoId(
                micItem
              )}">
                  <span>用户流id：${micItem.streamId}</span>
                  <span>昵称：${
                    micItem.isSelf
                      ? micItem.nickname + "(我)"
                      : micItem.nickname
                  }</span>
                  <span class="demo-connect-mic-list-item__info__video">摄像头状态：${
                    micItem.isVideoMuted ? "关闭" : "开启"
                  }</span>
                  <span class="demo-connect-mic-list-item__info__audio">麦克风状态：${
                    micItem.isAudioMuted ? "关闭" : "开启"
                  }</span>
                  ${
                    micItem.isLocal
                      ? `<span class="demo-connect-mic-list-item__info__mirror">镜像状态：${
                          micItem.mirrorEnabled ? "开启" : "关闭"
                        }</span>`
                      : ""
                  }
                </div>
            </div>
            `
          )
          .join("");

        // 注意：这里延迟去推流和订阅流，因为有可能渲染的元素还没插入到 DOM 中，并且 connectMicList 会变化，所以这里延迟一下
        setTimeout(() => {
          connectMicList.forEach((micItem) => {
            const $micItem = document.getElementById(generateMicId(micItem));
            if (!$micItem) {
              console.warn("找不到对应的元素", micItem);
              return;
            }

            if (micItem.isLocal) {
              micItem.publishStream({
                element: $micItem,
```

```
            control: false, // 是否显示自带的控制按钮，默认 false
          });
        } else {
          micItem.subscribeStream({
            element: $micItem,
            control: false,
          });
        }
      });
    }, 300);
  }

  /**
   * 绑定 UI 事件
   */
  function bindUIEvent(watchCore) {
    bindPlayerUIEvent(watchCore);
    // 设备检测
    const $openSettingBtn = document.querySelector(
      ".demo-connect-mic-control__open-setting-btn"
    );
    $openSettingBtn.addEventListener("click", () => {
      if (!checkConnectMicEnabled()) return;

      watchCore.connectMic.openDeviceSetting();
    });

    // 申请连麦
    const $applyConnectMicBtn = document.querySelector(
      ".demo-connect-mic-control__apply-btn"
    );
    $applyConnectMicBtn.addEventListener("click", async () => {
      if (!checkConnectMicEnabled()) return;

      if (!checkApplyConnectMicEnabled()) {
        console.warn("请先等讲师发起连麦！");
        return;
      }

      if (checkIsConnectMicing()) {
        console.warn("当前已在连麦中，请勿重复申请！");
        return;
      }

      const connectMicType = globalStore.connectMicInfo.connectMicType;

      console.warn(
        "正在申请连麦，当前连麦类型为：",
        connectMicType === ConnectMicType.Video ? "视频连麦" : "音频连麦"
      );

      const result = await watchCore.connectMic.applyConnectMic();
      if (
        !result.success &&
        result.failReason === ConnectMicError.GetDevicePermissionFail
      ) {
```

```
    alert("无法获取设备权限或无设备，请检查后重试！");
    return;
  }
  if (result.success) {
    console.warn("申请连麦成功，请等待讲师同意上麦");
  } else {
    console.warn("申请连麦失败", result.failReason);
  }
});

// 取消申请连麦
const $applyCancelBtn = document.querySelector(
  ".demo-connect-mic-control__cancel-apply-btn"
);
$applyCancelBtn.addEventListener("click", () => {
  if (!checkConnectMicEnabled()) return;

  if (!checkApplyConnectMicEnabled()) {
    console.warn("请先等讲师发起连麦！");
    return;
  }

  if (checkIsConnectMicing()) {
    console.warn("当前已在连麦中，无法取消申请，请结束连麦！");
    return;
  }

  watchCore.connectMic.cancelApplyConnectMic();
  console.warn("取消申请连麦成功");
});

// 结束连麦
const $hangUpBtn = document.querySelector(
  ".demo-connect-mic-control__hang-up-btn"
);
$hangUpBtn.addEventListener("click", () => {
  if (!checkConnectMicEnabled()) return;

  if (!checkIsConnectMicing()) {
    console.warn("当前不在连麦中，无法结束连麦！");
    return;
  }

  watchCore.connectMic.endConnectMic();
  console.warn("结束连麦成功");
});

// 切换摄像头开关
const $videoToggleBtn = document.querySelector(
  ".demo-connect-mic-control__video-toggle"
);
$videoToggleBtn.addEventListener("click", () => {
  if (!checkConnectMicEnabled()) return;

  const connectMicType = globalStore.connectMicInfo.connectMicType;
  if (connectMicType === ConnectMicType.Audio) {
```

```
    alert("当前为音频连麦，无法切换摄像头");
    return;
  }

  const localMicItem = getLocalMicItem();
  if (!localMicItem) {
    alert("当前未连麦，无法切换摄像头");
    return;
  }

  if (localMicItem.isVideoMuted) {
    console.log("开启摄像头");
    watchCore.connectMic.enabledVideo();
  } else {
    console.log("关闭摄像头");
    watchCore.connectMic.disabledVideo();
  }
});

// 切换麦克风开关
const $audioToggleBtn = document.querySelector(
  ".demo-connect-mic-control__audio-toggle"
);
$audioToggleBtn.addEventListener("click", () => {
  if (!checkConnectMicEnabled()) return;

  const localMicItem = getLocalMicItem();
  if (!localMicItem) {
    alert("当前未连麦，无法切换麦克风");
    return;
  }

  if (localMicItem.isAudioMuted) {
    console.log("开启麦克风");
    watchCore.connectMic.enabledAudio();
  } else {
    console.log("关闭麦克风");
    watchCore.connectMic.disabledAudio();
  }
});

// 切换镜像
const $mirrorToggleBtn = document.querySelector(
  ".demo-connect-mic-control__mirror-toggle"
);
$mirrorToggleBtn.addEventListener("click", () => {
  if (!checkConnectMicEnabled()) return;

  const connectMicType = globalStore.connectMicInfo.connectMicType;
  if (connectMicType === ConnectMicType.Audio) {
    alert("当前为音频连麦，无法切换镜像");
    return;
  }

  const localMicItem = getLocalMicItem();
  if (!localMicItem) {
```

```
    alert("当前未连麦，无法切换镜像");
    return;
  }

  const mirrorEnabled = watchCore.connectMic.getMirrorEnabled();
  watchCore.connectMic.toggleMirror();
  const newMirrorEnabled = !mirrorEnabled;
  console.log(
    mirrorEnabled ? "关闭镜像" : "开启镜像",
    "当前镜像状态：",
    newMirrorEnabled
  );

  // 更新镜像状态显示
  if (localMicItem) {
    renderMicInfoMirrorText({
      mirrorEnabled: newMirrorEnabled,
      micItem: localMicItem,
    });
  }
});

// 接受邀请上麦
const $acceptInviteBtn = document.querySelector(
  ".demo-connect-mic-control__accept-invite-btn"
);
$acceptInviteBtn.addEventListener("click", () => {
  if (!checkConnectMicEnabled()) return;
  watchCore.connectMic.acceptInvite();
  console.log("已接受邀请上麦");
});

// 拒绝邀请上麦
const $refuseInviteBtn = document.querySelector(
  ".demo-connect-mic-control__refuse-invite-btn"
);
$refuseInviteBtn.addEventListener("click", () => {
  if (!checkConnectMicEnabled()) return;
  watchCore.connectMic.refuseInvite();
  console.log("已拒绝邀请上麦");
});

// 开启本地预览
const $startPreviewBtn = document.querySelector(
  ".demo-connect-mic-control__start-preview-btn"
);
$startPreviewBtn.addEventListener("click", async () => {
  if (!checkConnectMicEnabled()) return;
  const previewEl = document.getElementById("preview-container");
  try {
    await watchCore.connectMic.startPreview({
      videoEl: previewEl,
      video: true,
      audio: true,
    });
    console.log("本地预览已开启");
```

```
    } catch (e) {
      console.error("开启本地预览失败", e);
    }
  });

  // 停止本地预览
  const $stopPreviewBtn = document.querySelector(
    ".demo-connect-mic-control__stop-preview-btn"
  );
  $stopPreviewBtn.addEventListener("click", () => {
    if (!checkConnectMicEnabled()) return;
    watchCore.connectMic.stopPreview();
    console.log("本地预览已停止");
  });

  // 切换本地音视频开关
  const $toggleLocalMuteBtn = document.querySelector(
    ".demo-connect-mic-control__toggle-local-mute-btn"
  );
  $toggleLocalMuteBtn.addEventListener("click", () => {
    if (!checkConnectMicEnabled()) return;
    const current = watchCore.connectMic.getLocalMuteStatus();
    watchCore.connectMic.setLocalMuteStatus({
      video: !current.video,
      audio: !current.audio,
    });
    console.log("切换本地音视频开关");
  });
}

/**
 * 绑定连麦模块相关事件
 */
function bindConnectMicEvent(watchCore) {
  // 连麦模块-监听连麦信息改变
  watchCore.connectMic.eventEmitter.on(
    ConnectMicEvents.ConnectMicInfoChange,
    () => {
      globalStore.connectMicInfo =
        watchCore.connectMic.getConnectMicInfo();

      document.getElementById("connect-mic-status").innerText =
        globalStore.connectMicInfo.connectMicStatus;
      document.getElementById("open-mic-status").innerText = globalStore
        .connectMicInfo.openMicStatus
        ? "开启"
        : "关闭";

      // 更新镜像状态显示
      const localMicItem = getLocalMicItem();
      if (localMicItem) {
        renderMicInfoMirrorText({
          mirrorEnabled: globalStore.connectMicInfo.mirrorEnabled,
          micItem: localMicItem,
        });
      }
```

```
  }
);

// 连麦模块-监听网络信息改变
watchCore.connectMic.eventEmitter.on(
  ConnectMicEvents.NetworkInfoChange,
  () => {
    globalStore.networkInfo = watchCore.connectMic.getNetworkInfo();
  }
);

// 连麦模块-监听设备采集失败
watchCore.connectMic.eventEmitter.on(
  ConnectMicEvents.DeviceRecoverFail,
  () => {
    alert("自动采集失败，请检查设备正常连接后刷新页面重新进入");
  }
);

// 连麦模块-监听邀请上麦
watchCore.connectMic.eventEmitter.on(
  ConnectMicEvents.InviteConnectMic,
  () => {
    console.warn("讲师邀请你上麦！");
    watchCore.connectMic.openInviting();
  }
);

// 连麦模块-监听邀请上麦倒计时
watchCore.connectMic.eventEmitter.on(
  ConnectMicEvents.InviteCountDown,
  (data) => {
    document.getElementById("invite-count-down").innerText =
      `剩余 ${data.remain}s / 总 ${data.total}s`;
  }
);

// 连麦模块-监听本地音视频开关变更
watchCore.connectMic.eventEmitter.on(
  ConnectMicEvents.LocalMuteStatusChange,
  (data) => {
    document.getElementById("local-mute-status").innerText =
      `摄像头：${data.video ? "关闭" : "开启"}，麦克风：${data.audio ? "关闭" : "开启"}`;
  }
);

// 连麦模块-连麦列表修改
watchCore.connectMic.eventEmitter.on(
  ConnectMicEvents.ConnectMicListChange,
  () => {
    globalStore.connectMicList =
      watchCore.connectMic.getConnectMicList();
    renderConnectMicList();
  }
);
```

```
  // 连麦模块-监听讲师挂断当前连麦者
  watchCore.connectMic.eventEmitter.on(
    ConnectMicEvents.TeacherHangUp,
    () => {
      alert("讲师已挂断你的连麦");
    }
  );

  // 连麦模块-监听连麦人数到达上限
  watchCore.connectMic.eventEmitter.on(
    ConnectMicEvents.ConnectMicOverLimit,
    () => {
      alert("连麦失败，连麦人数已到达上限");
    }
  );

  // 连麦模块-监听本地流的摄像头状态变化
  watchCore.connectMic.eventEmitter.on(
    ConnectMicEvents.LocalVideoMuteChange,
    ({ isVideoMuted }) => {
      renderMicInfoVideoText({
        isVideoMuted,
        micItem: getLocalMicItem(),
      });
    }
  );

  // 连麦模块-监听本地流的麦克风状态变化
  watchCore.connectMic.eventEmitter.on(
    ConnectMicEvents.LocalAudioMuteChange,
    ({ isAudioMuted }) => {
      renderMicInfoAudioText({
        isAudioMuted,
        micItem: getLocalMicItem(),
      });
    }
  );

  // 连麦模块-监听远端流的摄像头状态变化
  watchCore.connectMic.eventEmitter.on(
    ConnectMicEvents.RemoteVideoMuteChange,
    ({ micItem, isVideoMuted }) => {
      renderMicInfoVideoText({ isVideoMuted, micItem });
    }
  );

  // 连麦模块-监听远端流的麦克风状态变化
  watchCore.connectMic.eventEmitter.on(
    ConnectMicEvents.RemoteAudioMuteChange,
    ({ micItem, isAudioMuted }) => {
      renderMicInfoAudioText({ isAudioMuted, micItem });
    }
  );
}

/**
```

```
 * 绑定 watch-sdk 事件
 */
function bindWatchSdkEvent(watchCore) {
  // 监听核心 - setup 钩子
  watchCore.eventEmitter.on(PolyvWatchCoreEvents.WatchCoreSetuped, () => {
    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);
    }
  );

  bindConnectMicEvent(watchCore);
}

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

/**
 * 安装连麦功能
 */
async function setupConnectMic(watchCore) {
  const result = await watchCore.connectMic.setupConnectMic();

  if (result.success) {
    globalStore.connectMicInited = true;
    console.warn("正常初始化连麦功能");
  } else {
    console.log("连麦功能初始化失败", result.failReason);
  }
}

/**
 * 安装核心实例和授权
 */
async function setupWatchCore(watchCore) {
  await watchCore.setup();
```

```
        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=classmethoddoc_playermodule_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
        });
      }

      /**
       * 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: "", // 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
        await watchCore.connect();

        // 创建播放器
        await createPlayer(watchCore);

        // 需要等聊天室连接完成后，再加载连麦功能
        setupConnectMic(watchCore);
      }

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