DEVICES SETUP

1. List audio input, audio ouput, and video input devices

```
const audioInputDevices = await
meetingSession.audioVideo.listAudioInputDevices();
const audioOutputDevices = await
meetingSession.audioVideo.listAudioOutputDevices();
const videoInputDevices = await
meetingSession.audioVideo.listVideoInputDevices();

// An array of MediaDeviceInfo objects
audioInputDevices.forEach((mediaDeviceInfo) => {
  console.log(
    `Device ID: ${mediaDeviceInfo.deviceId} Microphone:
${mediaDeviceInfo.label}`
  );
});
```

2. Choose audio input and audio output devices by passing the deviceId of a `MediaDeviceInfo` object.

```
const audioInputDeviceInfo = /* An array item from
meetingSession.audioVideo.listAudioInputDevices */;
await
meetingSession.audioVideo.chooseAudioInputDevice(audioInputDeviceInfo.devi
ceId);

const audioOutputDeviceInfo = /* An array item from
meetingSession.audioVideo.listAudioOutputDevices */;
await
meetingSession.audioVideo.chooseAudioOutputDevice(audioOutputDeviceInfo.de
viceId);
```

3. Choose a video input device by passing the `deviceId` of a `MediaDeviceInfo` object.

```
const videoInputDeviceInfo = /* An array item from
meetingSession.audioVideo.listVideoInputDevices */;
await
meetingSession.audioVideo.chooseVideoInputDevice(videoInputDeviceInfo.devi
ceId);
// You can pass null to choose none. If the previously chosen camera has
an LED light on,
// it will turn off indicating the camera is no longer capturing.
await meetingSession.audioVideo.chooseVideoInputDevice(null);
```

4. Add a device change `observer` to receive the `updated device` list.

```
const observer = {
  audioInputsChanged: (freshAudioInputDeviceList) => {
    // An array of MediaDeviceInfo objects
    freshAudioInputDeviceList.forEach((mediaDeviceInfo) => {
      console.log(
        `Device ID: ${mediaDeviceInfo.deviceId} Microphone:
${mediaDeviceInfo.label}`
      );
    });
  },
  audioOutputsChanged: (freshAudioOutputDeviceList) => {
    console.log("Audio outputs updated: ", freshAudioOutputDeviceList);
  },
  videoInputsChanged: (freshVideoInputDeviceList) => {
    console.log("Video inputs updated: ", freshVideoInputDeviceList);
  },
};

meetingSession.audioVideo.addDeviceChangeObserver(observer);
```

STARTING A SESSION

- You need to bind a device and stream to an
- Once the session has started, you can talk and listen to attendees

```
const audioElement = /* HTMLAudioElement object e.g.
document.getElementById('audio-element-id') */;
meetingSession.audioVideo.bindAudioElement(audioElement);

const observer = {
  audioVideoDidStart: () => {
    console.log('Started');
  }
};
meetingSession.audioVideo.addObserver(observer);
meetingSession.audioVideo.start();
```

AUDIO

- Mute and unmute an audio input.

```
// Mute
meetingSession.audioVideo.realtimeMuteLocalAudio();
// Unmute
const unmuted = meetingSession.audioVideo.realtimeUnmuteLocalAudio();
```

```
if (unmuted) {
  console.log('Other attendees can hear your audio');
} else {
  // See the realtimeSetCanUnmuteLocalAudio use case below.
  console.log('You cannot unmute yourself');
}
```

- To check whether the local microphone is muted

```
const muted = meetingSession.audioVideo.realtimeIsLocalAudioMuted();
if (muted) {
  console.log('You are muted');
} else {
  console.log('Other attendees can hear your audio');
}
```

- Prevent users from unmuting themselves

```
meetingSession.audioVideo.realtimeSetCanUnmuteLocalAudio(false);
// Optional: Force mute.
meetingSession.audioVideo.realtimeMuteLocalAudio();

const unmuted = meetingSession.audioVideo.realtimeUnmuteLocalAudio();
console.log(`${unmuted} is false. You cannot unmute yourself`)
```

VIDEO

- In chime SDK terms, a video tile is an object containing an attendee ID, a video stream, etc.
- To view a video in your application, you must bind a tile to a `<video>` element.

> - Make sure you bind a tile to the same video element until the tile is removed.
> - A local video tile can be identified using localTile property.
> - A tile is created with a new tile ID when the same remote attendee restarts the video.

- Share local video

```
const videoElement = /* HTMLVideoElement object e.g.
document.getElementById('video-element-id') */;
// Make sure you have chosen your camera. In this use case, you will
choose the first device.
const videoInputDevices = await
meetingSession.audioVideo.listVideoInputDevices();
// The camera LED light will turn on indicating that it is now capturing.
// See the "Device" section for details.
await
```

```
meetingSession.audioVideo.chooseVideoInputDevice(videoInputDevices[0].devi
ceId);
const observer = {
  // videoTileDidUpdate is called whenever a new tile is created or
tileState changes.
  videoTileDidUpdate: tileState => {
    // Ignore a tile without attendee ID and other attendee's tile.
    if (!tileState.boundAttendeeId || !tileState.localTile) {
      return;
    }
    meetingSession.audioVideo.bindVideoElement(tileState.tileId,
videoElement);
  }
};
meetingSession.audioVideo.addObserver(observer);
meetingSession.audioVideo.startLocalVideoTile();
```

- Stop share local video

```
const videoElement = /* HTMLVideoElement object e.g.
document.getElementById('video-element-id') */;
let localTileId = null;
const observer = {
  videoTileDidUpdate: tileState => {
    // Ignore a tile without attendee ID and other attendee's tile.
    if (!tileState.boundAttendeeId || !tileState.localTile) {
      return;
    }
    // videoTileDidUpdate is also invoked when you call
startLocalVideoTile or tileState changes.
    // The tileState.active can be false in poor Internet connection, when
the user paused the video tile, or when the video tile first arrived.
    console.log(`If you called stopLocalVideoTile, ${tileState.active} is
false.`);
    meetingSession.audioVideo.bindVideoElement(tileState.tileId,
videoElement);
    localTileId = tileState.tileId;
  },
  videoTileWasRemoved: tileId => {
    if (localTileId === tileId) {
      console.log(`You called removeLocalVideoTile. videoElement can be
bound to another tile.`);
      localTileId = null;
    }
  }
};
meetingSession.audioVideo.addObserver(observer);
meetingSession.audioVideo.stopLocalVideoTile();
// Optional: You can remove the local tile from the session.
meetingSession.audioVideo.removeLocalVideoTile();
```