desktopCapturer

> *Access information about media sources that can be used to capture audio and video from the desktop using the `navigator.mediaDevices.getUserMedia` API.*

Process: Main

The following example shows how to capture video from a desktop window whose title is `Electron`:

```
// In the main process.
const { desktopCapturer } = require('electron')

desktopCapturer.getSources({ types: ['window', 'screen'] }).then(async sources => {
  for (const source of sources) {
    if (source.name === 'Electron') {
      mainWindow.webContents.send('SET_SOURCE', source.id)
      return
    }
  }
})
```

```
// In the preload script.
const { ipcRenderer } = require('electron')

ipcRenderer.on('SET_SOURCE', async (event, sourceId) => {
  try {
    const stream = await navigator.mediaDevices.getUserMedia({
      audio: false,
      video: {
        mandatory: {
          chromeMediaSource: 'desktop',
          chromeMediaSourceId: sourceId,
          minWidth: 1280,
          maxWidth: 1280,
          minHeight: 720,
          maxHeight: 720
        }
      }
    })
    handleStream(stream)
  } catch (e) {
    handleError(e)
  }
})

function handleStream (stream) {
  const video = document.querySelector('video')
  video.srcObject = stream
  video.onloadedmetadata = (e) => video.play()
}
```

```
function handleError (e) {
  console.log(e)
}
```

To capture video from a source provided by `desktopCapturer` the constraints passed to [`navigator.mediaDevices.getUserMedia`]() must include `chromeMediaSource: 'desktop'`, and `audio: false`.

To capture both audio and video from the entire desktop the constraints passed to [`navigator.mediaDevices.getUserMedia`]() must include `chromeMediaSource: 'desktop'`, for both `audio` and `video`, but should not include a `chromeMediaSourceId` constraint.

```
const constraints = {
  audio: {
    mandatory: {
      chromeMediaSource: 'desktop'
    }
  },
  video: {
    mandatory: {
      chromeMediaSource: 'desktop'
    }
  }
}
```

Methods

The `desktopCapturer` module has the following methods:

`desktopCapturer.getSources(options)`

- `options` Object
  - `types` string[] - An array of strings that lists the types of desktop sources to be captured, available types are `screen` and `window`.
  - `thumbnailSize` [Size]() (optional) - The size that the media source thumbnail should be scaled to. Default is `150` x `150`. Set width or height to 0 when you do not need the thumbnails. This will save the processing time required for capturing the content of each window and screen.
  - `fetchWindowIcons` boolean (optional) - Set to true to enable fetching window icons. The default value is false. When false the appIcon property of the sources return null. Same if a source has the type screen.

Returns `Promise<DesktopCapturerSource[]>` - Resolves with an array of [`DesktopCapturerSource`]() objects, each `DesktopCapturerSource` represents a screen or an individual window that can be captured.

Note Capturing the screen contents requires user consent on macOS 10.15 Catalina or higher, which can detected by [`systemPreferences.getMediaAccessStatus`]().

Caveats

`navigator.mediaDevices.getUserMedia` does not work on macOS for audio capture due to a fundamental limitation whereby apps that want to access the system's audio require a signed kernel extension. Chromium, and by extension Electron, does not provide this.

It is possible to circumvent this limitation by capturing system audio with another macOS app like Soundflower and passing it through a virtual audio input device. This virtual device can then be queried with `navigator.mediaDevices.getUserMedia`.