

Choose Cameras, Microphones and Speakers from Your Web App



By Sam Dutton

Sam is a Developer Advocate

Modern browsers make it possible to select input and output devices including cameras, microphones and speakers.

For example:

- On a phone, select the front or rear-facing camera.
- On a laptop, choose the internal speakers or a speaker connected by Bluetooth.
- For a video chat, choose internal or external microphone or camera.

All this functionality is exposed by the `MediaDevices` object, which is returned by `navigator.mediaDevices`.

`MediaDevices` has two methods, both implemented in Chrome 47 on desktop and Android: `enumerateDevices()` and `getUserMedia()`.

Select audio and video X

https://webrtc.github.io/samples/src/content/devices/input-output/

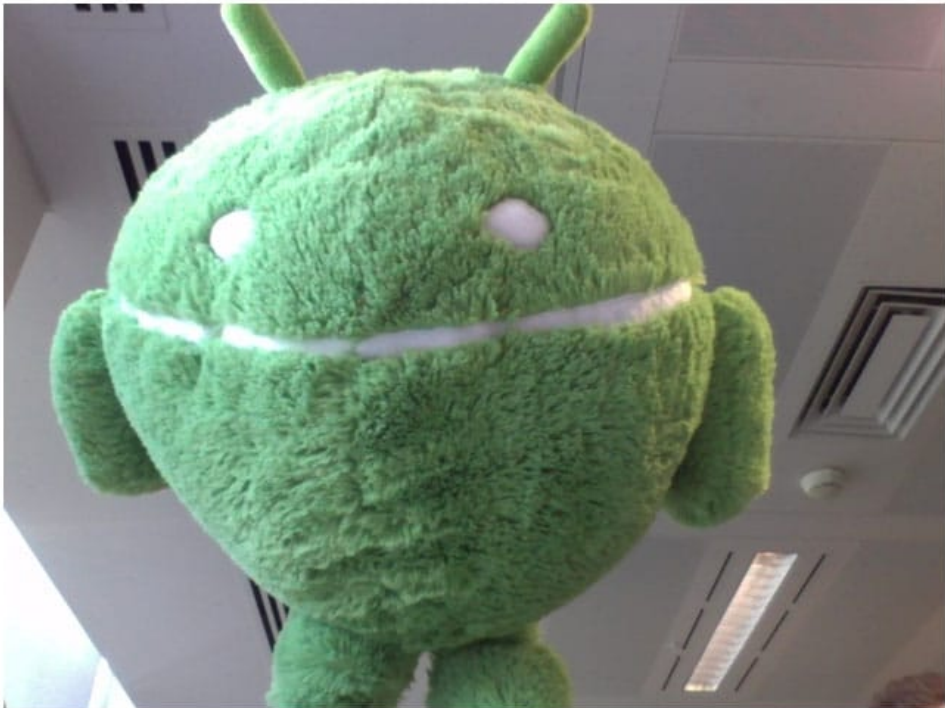
WebRTC samples Select sources & outputs

Get available audio, video sources and audio output devices* from `mediaDevices.enumerateDevices()` then set the source for `getUserMedia()` using a `deviceId` constraint.

Audio input source: Default

Audio output destination: ☒ Default
Bluetooth Speaker
Built-in Output

Video source: FaceTime HD Camera (Built-in) (05ac:8510)



[View source on GitHub](#)

`enumerateDevices()`

Returns a Promise giving access to an array of `MediaDeviceInfo` objects for available devices.

The method is similar to `MediaStreamTrack.getSources()` but unlike that method (which was only ever implemented in Chrome) it's standards compliant and includes audio output devices. You can try this out with the demos below.

Here's some slightly simplified code from one of the demos:

```
navigator.mediaDevices.enumerateDevices()
  .then(gotDevices)
  .catch(errorCallback);
...
function gotDevices(deviceInfos) {

  ...

  for (var i = 0; i !== deviceInfos.length; ++i) {
    var deviceInfo = deviceInfos[i];
    var option = document.createElement('option');
    option.value = deviceInfo.deviceId;
    if (deviceInfo.kind === 'audioinput') {
      option.text = deviceInfo.label ||
        'Microphone ' + (audioInputSelect.length + 1);
      audioInputSelect.appendChild(option);
    } else if (deviceInfo.kind === 'audiooutput') {
      option.text = deviceInfo.label || 'Speaker ' +
        (audioOutputSelect.length + 1);
      audioOutputSelect.appendChild(option);
    } else if (deviceInfo.kind === 'videoinput') {
      option.text = deviceInfo.label || 'Camera ' +
        (videoSelect.length + 1);
      videoSelect.appendChild(option);
    }

    ...

  }
}
```



Having retrieved the IDs of available devices with `enumerateDevices()`, you can use `setSinkId()` (defined in the [Audio Output Devices API](#)) to change the audio output destination for a video or audio element:

```
element.setSinkId(sinkId)
  .then(function() {
    console.log('Audio output device attached: ' + sinkId);
  })
  .catch(function(error) {
    // ...
  });
```



This method sets the output device for audio from the element. Once `setSinkId()` has been called, you can get the ID of the current output audio device for the element with the `sinkId` property.

getUserMedia()

This replaces `navigator.getUserMedia()`, but instead of using a callback, returns a Promise that gives access to a `MediaStream`. Developers are encouraged to use `MediaDevices.getUserMedia()`, but there's no plan to remove `navigator.getUserMedia()`: it remains part of the spec.

There's a demo at the [WebRTC samples site](#).

Here's a fragment of code from the demo:

```
navigator.mediaDevices.getUserMedia(constraints)
  .then(function(stream) {
    var videoTracks = stream.getVideoTracks();
    console.log('Got stream with constraints:', constraints);
    console.log('Using video device: ' + videoTracks[0].label);
    stream.onended = function() {
      console.log('Stream ended');
    };
    window.stream = stream; // make variable available to console
    video.srcObject = stream;
  })
  .catch(function(error) {
    // ...
  })
```



Flag waiving

The `enumerateDevices()` method is 'flagless' in Chrome, whereas for `MediaDevices.getUserMedia()` you still need to enable **Experimental Web Platform features** in `chrome://flags` or use the following command line flag:

```
--enable-blink-features=GetUserMedia
```



Likewise for `setSinkId()`: enable **Experimental Web Platform features** or use a flag:

```
--enable-blink-features=AudioOutputDevices
```



More details about browser support below.

The future

The proposed `ondevicechange` event handler does what it says: the `devicechange` event is fired when the set of available devices changes, and in a handler you can call `enumerateDevices()` to get the new list of devices. This hasn't been implemented in any browser yet.

The [Screen Capture draft](#) is an extension to the Media Capture API which proposes a `MediaDevices.getDisplayMedia()` method that enables regions of a user's display to be used as the source of a media stream. There is also a `MediaDevices` extension proposal for [getSupportedConstraints\(\)](#), which provides information about what constraints could be used for a `getUserMedia()` call: audio and video capabilities supported by the browser.

Demos

- [getUserMedia\(\)](#). [↗](#)
- `enumerateDevices()`:
 - [Select sources & outputs](#)
 - [Output device selection](#) [↗](#)
- [MediaDevices shim](#)

Find out more

- [Mozilla Developer Network: Media Devices](#)
- [Implementation status](#)
- Media Capture and Streams Editor's Draft: [MediaDevices](#)
- [Audio Output Devices API](#)

Except as otherwise noted, the content of this page is licensed under the [Creative Commons Attribution 3.0 License](#), and code samples are licensed under the [Apache 2.0 License](#). For details, see our [Site Policies](#). Java is a registered trademark of Oracle and/or its affiliates.

Last updated July 2, 2018.