Record Audio and Video with MediaRecorder

By Sam Dutton

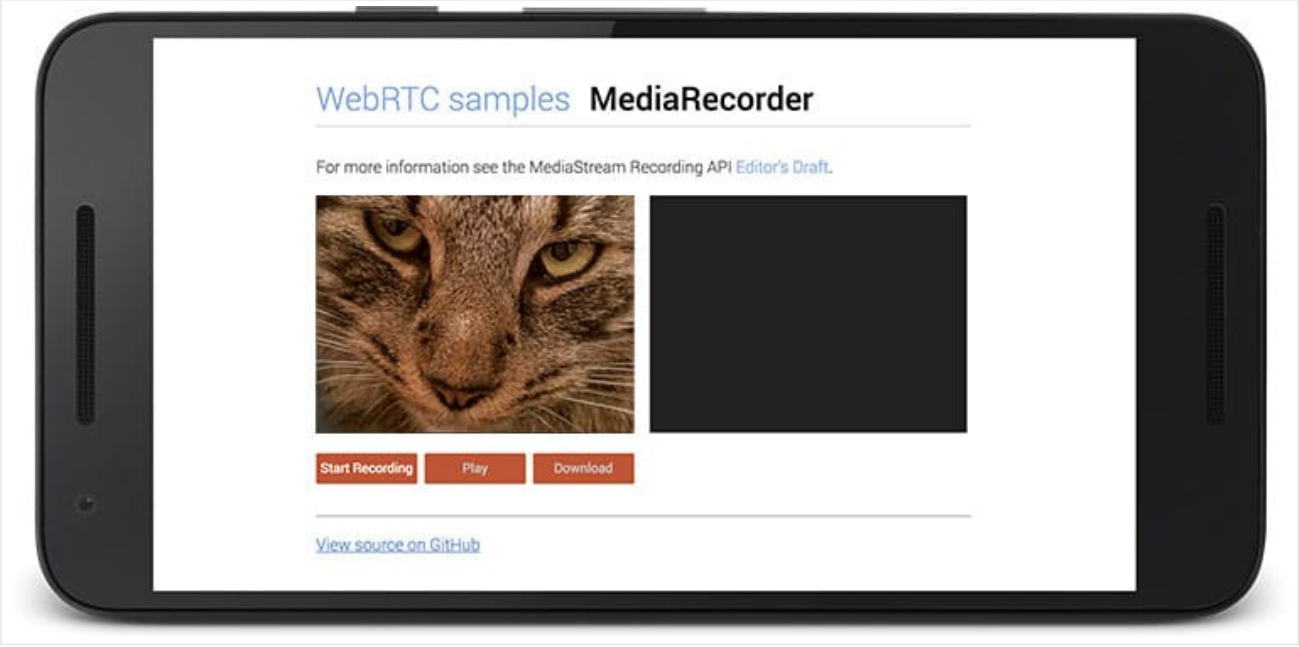
Sam is a Developer Advocate

Break out the champagne and doughnuts! The most starred Chrome feature EVER has now been implemented.

Imagine a ski-run recorder that synchronizes video with GeoLocation data, or a super-simple voice memo app, or a widget that enables you to record a video and upload it to YouTube — all without plugins.

The MediaRecorder API enables you to record audio and video from a web app. It's available now in Firefox and in Chrome for Android and desktop.

Try it out here.



A word about support:

• To use MediaRecorder in Chrome 47 and 48, enable **experimental Web Platform features** from the chrome://flags page.

• Audio recording work in Firefox and in Chrome 49 and above; Chrome 47 and 48 only support video recording.

• In Chrome on Android you can save and download recordings made with MediaRecorder, but it's not yet possible to view a recording in a video element via `window.URL.createObjectURL()`. See this bug.

The API is straightforward, which I'll demonstrate using code from the WebRTC sample repo demo. Note that the API can only be used from secure origins only: HTTPS or localhost.

First up, instantiate a MediaRecorder with a MediaStream. Optionally, use an `options` parameter to specify the desired output format:

```
var options = {mimeType: 'video/webm; codecs=vp9'};
mediaRecorder = new MediaRecorder(stream, options);
```

The MediaStream can be from:

- A `getUserMedia()` call.
- The receiving end of a WebRTC call.
- A screen recording.
- Web Audio, once this issue is implemented.

For `options` it's possible to specify the MIME type and, in the future, audio and video bitrates.

MIME types have more or less specific values, combining container and codecs. For example:

- audio/webm
- video/webm
- video/webm;codecs=vp8
- video/webm;codecs=vp9

Use the static method `MediaRecorder.isTypeSupported()` to check if a MIME type is supported, for example when you instantiate MediaRecorder:

```
var options;
if (MediaRecorder.isTypeSupported('video/webm;codecs=vp9')) {
  options = {mimeType: 'video/webm; codecs=vp9'};
} else if (MediaRecorder.isTypeSupported('video/webm;codecs=vp8')) {
   options = {mimeType: 'video/webm; codecs=vp8'};
} else {
  // ...
}
```

The full list of MIME types supported by MediaRecorder in Chrome is available here.

Caution: Instantiation will fail if the browser doesn't support the MIME type specified, so use `MediaRecorder.isTypeSupported()` or try/catch — or leave out the `options` argument if you're happy with the browser default.

Next, add a data handler and call the `start()` method to begin recording:

```
var recordedChunks = [];

var options = {mimeType: 'video/webm;codecs=vp9'};
mediaRecorder = new MediaRecorder(stream, options);
mediaRecorder.ondataavailable = handleDataAvailable;
mediaRecorder.start();

function handleDataAvailable(event) {
  if (event.data.size > 0) {
    recordedChunks.push(event.data);
  } else {
    // ...
  }
}
```

This examples adds a Blob to the `recordedChunks` array whenever data becomes available. The `start()` method can optionally be given a `timeSlice` argument that specifies the length of media to capture for each Blob.

When you've finished recording, tell the MediaRecorder:

```
mediaRecorder.stop();
```

Play the recorded Blobs in a video element by creating a 'super-Blob' from the array of recorded Blobs:

```
function play() {
  var superBuffer = new Blob(recordedChunks);
  videoElement.src =
    window.URL.createObjectURL(superBuffer);
}
```

Alternatively, you could upload to a server via XHR, or use an API like YouTube (see the experimental demo below).

Download can be achieved with some link hacking:

```
function download() {
  var blob = new Blob(recordedChunks, {
    type: 'video/webm'
  });
  var url = URL.createObjectURL(blob);
  var a = document.createElement('a');
  document.body.appendChild(a);
  a.style = 'display: none';
  a.href = url;
  a.download = 'test.webm';
  a.click();
  window.URL.revokeObjectURL(url);
}
```

Feedback on the APIs and demos

The ability to record audio and video without plugins is relatively new to web apps, so we particularly appreciate your feedback on the APIs.

- MediaRecorder implementation bug: crbug.com/262211
- Chrome: crbug.com/new
- Firefox: bugzil.la
- Demos: github.com/webrtc/samples

We'd also like to know what usage scenarios are most important to you, and what features you would like us to prioritize. Comment on this article or track progress at crbug.com/262211.

Demos

- webrtc.github.io/samples/src/content/getusermedia/record
- simpl.info/mr (same code, easier URL for mobile!)
- Record a video and upload it to YouTube with an experimental custom <google-youtube-upload> element

Apps

- Paul Lewis's Voice Memos app now has MediaRecorder support, polyfilled for browsers that don't support MediaRecorder audio.

Polyfills

- Muaz Khan's MediaStreamRecorder is a JavaScript library for recording audio and video, compatible with MediaRecorder.
- Recorderjs enables recording from a Web Audio API node. You can see this in action in Paul Lewis's Voice Memos app.

Browser support

- Chrome 49 and above by default
- Chrome desktop 47 and 48 with Experimental Web Platform features enabled from chrome://flags
- Firefox from version 25
- Edge: 'Under Consideration'

Spec

w3c.github.io/mediacapture-record/MediaRecorder.html

API information

developer.mozilla.org/en/docs/Web/API/MediaRecorder_API

Last updated July 2, 2018.