MediaStream Recording API

Web technology for developers
Web APIs
MediaStream Recording API

Related Topics

MediaStream Recording API
Guides
Recording a media element
Using the MediaStream Recording API
Interfaces
MediaRecorder
MediaRecorderErrorEvent
BlobEvent
Events
start
stop
dataavailable
pause
resume
error

The **MediaStream Recording API**, sometimes simply referred to as the *Media Recording API* or the *MediaRecorder API*, is closely affiliated with the Media Capture and Streams API and the WebRTC API. The MediaStream Recording API makes it possible to capture the data generated by a `MediaStream` or `HTMLMediaElement` object for analysis, processing, or saving to disk. It's also surprisingly easy to work with.

Basic concepts

The MediaStream Recording API is comprised of a single major interface, `MediaRecorder`, which does all the work of taking the data from a `MediaStream` and delivering it to you for processing. The data is delivered by a series of `dataavailable` events, already in the format you specify when creating the `MediaRecorder`. You can then process the data further or write it to file as desired.

Overview of the recording process

The process of recording a stream is simple:

1. Set up a `MediaStream` or `HTMLMediaElement` (in the form of an `<audio>` or `<video>` element) to serve as the source of the media data.
2. Set `MediaRecorder.ondataavailable` to an event handler for the `dataavailable` event; this will be called whenever data is available for you.
3. Create a `MediaRecorder` object, specifying the source stream and any desired options (such as the container's MIME type or the desired bit rates of its tracks.
4. Once the source media is playing and you've reached the point where you're ready to record video, call `MediaRecorder.start()` to begin recording.
5. Your `dataavailable` event handler gets called every time there's data ready for you to do with as you will; the event has a `data` attribute whose value is a `Blob` that contains the media data. You can force a `dataavailable` event to occur, thereby delivering the latest sound to you so you can filter it, save it, or whatever.
6. Recording stops automatically when the source media stops playing.
7. You can stop recording at any time by calling `MediaRecorder.stop()`.

Note: Individual `Blob`s containing slices of the recorded media will not necessarily be individually playable. The media needs to be reassembled before playback.

If anything goes wrong during recording, an `error` event is sent to the `MediaRecorder`. You can listen for `error` events by setting up a `onerror` event handler.

Examining and controlling the recorder status

You can also use the properties of the `MediaRecorder` object to determine the state of the recording process, and its `pause()` and `resume()` methods to pause and resume recording of the source media.

If you need or want to check to see if a specific MIME type is supported, that's possible as well. Just call `MediaRecorder.isTypeSupported()`.

Examining potential input sources

If your goal is to record camera and/or microphone input, you may wish to examine the available input devices before beginning the process of constructing the `MediaRecorder`. To do so, you'll need to call `navigator.mediaDevices.enumerateDevices()` to get a list of the available media devices. You can then examine that list and identify the potential input sources, and even filter the list based on desired criteria.

In this code snippet, `enumerateDevices()` is used to examine the available input devices, locate those which are audio input devices, and create `<option>` elements that are then added to a `<select>` element representing an input source picker.

```
navigator.mediaDevices.enumerateDevices()
.then(function(devices) {
  devices.forEach(function(device) {
    let menu = document.getElementById("inputdevices");
    if (device.kind == "audioinput") {
      let item = document.createElement("option");
      item.innerHTML = device.label;
      item.value = device.deviceId;
      menu.appendChild(item);
    }
  });
});
```

Code similar to this can be used to let the user restrict the set of devices they wish to use.

For more information

To learn more about using the MediaStream Recording API, see Using the MediaStream Recording API, which shows how to use the API to record audio clips. A second article, Recording a media element, describes how to receive a stream from an `<audio>` or `<video>` element and use the captured stream (in this case, recording it and saving it to a local disk).

Reference

`BlobEvent`

Each time a chunk of media data is finished being recorded, it's delivered to consumers in `Blob` form using a `BlobEvent` of type `dataavailable`.

`MediaRecorder`

The primary interface that implements the MediaStream Recording API.

`MediaRecorderErrorEvent`

The interface that represents errors thrown by the MediaStream Recording API. Its `error` property is a `DOMException` that specifies that error occurred.

Specifications

Specification	Status	Comment
MediaStream Recording	W3C Working Draft	Initial definition

Browser compatibility

We're converting our compatibility data into a machine-readable JSON format. This compatibility table still uses the old format, because we haven't yet converted the data it contains. Find out how you can help!

Desktop | Mobile

Feature	Chrome	Firefox (Gecko)	Internet Explorer	Microsoft Edge	Opera	Safari (WebKit)
Basic support	47.0	25.0 (25.0)	No support	?	No support	No support

[1] The initial Firefox OS implementation only supported audio recording.

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

[3] Audio recording works in Chrome 49 and above; Chrome 47 and 48 only support video recording.

See also

- Using the MediaStream Recording API
- Recording a media element
- simpl.info MediaStream Recording demo, Sam Dutton
- `navigator.mediaDevices.getUserMedia()`
- HTML5's Media Recorder API in Action on Chrome and Firefox
- MediaRecorder polyfill for Safari and Edge
- TutorRoom: HTML5 video capture/playback/download using getUserMedia and the MediaRecorder API (source on GitHub)
- FingerSpell: Sign Language Fingerspelling practice using getUserMedia and the MediaRecorder API to create and download recordings, MediaRecorder API supported desktop browsers only (source on GitHub)
- Simple video recording demo
- Advanced media stream recorder sample
- OpenLang: HTML5 video language lab web application using MediaDevices and the MediaStream Recording API for video recording (source on GitHub)

Tags: API Audio Media Media Capture and Streams MediaStream Recording MediaStream Recording API Overview Reference Video

Contributors to this page: Sheppy, iskin, lazarjubenovic, SOCSIELEARNING, Ambar, dimaspirit

Last updated by: Sheppy, Jan 26, 2018, 10:52:04 AM