

# MediaStream Deprecations



By Sam Dutton

Sam is a Developer Advocate

If you work with `getUserMedia()` or WebRTC, you may need to adjust your code for Chrome 45 and later.

The MediaStream API represents synchronized streams of media. For example, a stream taken from camera and microphone input has synchronized video and audio tracks. Each track is represented by a MediaStreamTrack. (Not to be confused with the <track> element [\[?\]](#)!)

There are three `MediaStream` deprecations in Chrome 45:

- `MediaStream.ended`
- `MediaStream.label`
- `MediaStream.stop()`

In parallel are two additions:

- `MediaStream.active`
- `MediaStreamTrack.stop()`

These require the following changes:

- Use `MediaStream.active` to check if a `MediaStream` is streaming, not `MediaStream.ended`.
- Use `MediaStreamTrack.stop()` to stop streaming, not `MediaStream.stop()`.
- If you need a unique identifier for a `MediaStream` use `MediaStream.id` instead of `MediaStream.label`. `MediaStreamTrack.label` provides a human-readable name for the source device for a stream, e.g. *FaceTime HD Camera (Built-in) (05ac:8510)*.

You can see these in action: open [simpl.info/gum](http://simpl.info/gum) in Chrome (on a device with a camera) and view the Chrome DevTools console. The `MediaStream` object `stream` passed to the `getUserMedia()` callback in this demo is in global scope, so you can inspect it from the console. Call `stream.getTracks()[0]` to view the `MediaStreamTrack` for this stream.

simpl.info getUserMedia

The `MediaStream` object `stream` passed to the `getUserMedia()` callback in this demo is in global scope, so you can inspect it from the console.

For more information, see [Capturing Audio & Video in HTML5](#) on HTML5 Rocks.

[View source on GitHub](#)

Console

<top frame>

Filter  ☐ Regex

☐ Hide network messages

All Errors Warnings Info Logs Debug Handled

```

MediaStream {}
  active: true
  ended: false
  id: "eStTDwUtQ1C1G7gayh22aNAg6R34FXm47Fzh"
  label: "eStTDwUtQ1C1G7gayh22aNAg6R34FXm47F..."
  onactive: null
  onaddtrack: null
  onended: null
  oninactive: null
  onremovetrack: null
  __proto__: MediaStream

stream.getTracks()[0]
MediaStreamTrack {}
  enabled: true
  id: "56ed72b9-d4bc-4c8a-98d3-78e5ce8420e0"
  kind: "video"
  label: "FaceTime HD Camera (Built-in) (05ac:8..."
  muted: false
  onended: null
  onmute: null
  onunmute: null
  readyState: "live"
  __proto__: MediaStreamTrack
    clone: function clone()
    constructor: function MediaStreamTrack()
      enabled: (...)
    get enabled: function ()
    set enabled: function ()
      id: (...)
    get id: function ()
      kind: (...)
    get kind: function ()
      label: (...)
    get label: function ()
      muted: (...)
    get muted: function ()
      onended: (...)
    get onended: function ()
    set onended: function ()
      onmute: (...)
    get onmute: function ()
    set onmute: function ()
      onunmute: (...)
    get onunmute: function ()
    set onunmute: function ()
      readyState: (...)
    get readyState: function ()
    stop: function stop()
    __proto__: EventTarget

```

## Stop(), ended and active

When the Media Capture and Streams W3C Working Group looked at the problem of what happens when you add new tracks to a `MediaStream`, and whether an empty `MediaStream` is ended, they realized that there was no sensible way to implement ended on a `MediaStream` (as in 'will never start again'). In other parts of HTML5 'ended' means 'this has ended and will never resume'. 'Active' carries no such implication: an inactive stream can become active

again, for instance if a new track is added to it. Rather than maintain a confusing attribute and function, the Working Group decided to remove it.

Here's an example of how to use 'MediaStream.active' to check the status of a stream:

```
var gumStream;

navigator.getUserMedia({audio: false, video: true},
  function(stream) {
    gumStream = stream;
    // ...
  },
  function(error) {
    console.log('getUserMedia() error', error);
  });

// ...

if (gumStream.active) {
  // do something with the stream
}
```



Removing `stop()` from `MediaStream` did not remove any real functionality: processes for detaching source devices and so on have to be done on `MediaStreamTrack` anyway. Use `stop()` on `MediaStreamTrack` instead:

```
navigator.getUserMedia({audio: false, video: true},
  function(stream) {
    // can also use getAudioTracks() or getVideoTracks()
    var track = stream.getTracks()[0]; // if only one media track
    // ...
    track.stop();
  },
  function(error){
    console.log('getUserMedia() error', error);
  });
```



## label

It turns out that nobody could quite figure out a use for this property!

`MediaStream.label` had been added to the first version of the spec, but nobody really knew what `label` was for. It was also unclear what happened to `label` when a stream was sent via `RTCPeerConnection`.

The W3C Working Group asked around, and nobody wanted it, so they removed it.

To reiterate: `MediaStream.id` provides a unique identifier for a `MediaStream` and `MediaStreamTrack.label` provides the name of the source of a stream, such as the type of camera or microphone.

More information about `MediaStream` and `MediaStreamTrack` is available from [Mozilla Developer Network](#), and HTML5 Rocks provides an excellent introduction to `getUserMedia()` in [Capturing Audio & Video](#) [↗](#).

As ever, we appreciate your feedback on changes to Chrome. You can follow the bugs for these deprecations ([here](#) and [here](#)) and find more discussion and detail in the [Intent to Implement](#).

---

*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.*