W3C WebRTC/MediaCapture WG Meeting

October 12, 2017
8 AM PDT

Chairs: Stefan Hakansson

Bernard Aboba

Harald Alvestrand (finally back!)

W3C WG IPR Policy

- This group abides by the W3C patent policy https://www.w3.org/Consortium/Patent-Policy-20040205
- Only people and companies listed at https://www.w3.org/2004/01/pp-impl/47318/status are allowed to make substantive contributions to the WebRTC specs

Welcome!

- Welcome to the interim meeting of the W3C WebRTC WG!
- During this meeting, we hope to make progress on issues arising from the CR review of webrtc-pc
- If time permits, we will also discuss Media Capture issues
- Editor’s Draft updates to follow meeting

Current Status of WebRTC-PC

- 100 open issues:
  - **3 blocking advancement to CR: our main focus today - once resolved we go to CR**
  - 24 editorial
  - 16 arising from test suite development
  - 10 question
  - 7 PR exists
  - 4 pending IETF actions
  - 1 enhancement

TPAC coming up

- Draft agenda:
  - https://www.w3.org/2011/04/webrtc/wiki/November_6_-_7_2017#Agenda
- We’re looking for feedback!

About this Virtual Meeting

Information on the meeting:

- Meeting info:
  - https://www.w3.org/2011/04/webrtc/wiki/October_12_2017
- Link to latest drafts:
  - https://rawgit.com/w3c/mediacapture-main/master/getusermedia.html
  - https://rawgit.com/w3c/webrtc-pc/master/webrtc.html
  - https://rawgit.com/w3c/webrtc-stats/master/webrtc-stats.html
- Link to Slides has been published on WG wiki
- Scribe? IRC http://irc.w3.org/ Channel: #webrtc
- The meeting is being recorded.
- WebEx info here

For Discussion Today

- CR blocking WebRTC-PC Issues
  - Issue 1178/PR 1623: Need to describe whe
DTLS transport objects are created/change
(Taylor)
  - Issue 1406/PR 1631: When ICE restart resu
connection to a new endpoint (Taylor)
  - Issue 1283/PR 1570: Centering, Scaling, Cr
(Stefan)
- Non-CR-blocking WebRTC-PC Issues
  - Issue 1625/PR 1632: RTCPriorityType unde
combines relative bitrate with QoS priority (Taylor)

For Discussion Today (cont’d)

- **Media Capture Issues**
    - **Issue 478: Content hints for MediaStreamTrack (Peter Boström)**
    - **Issue 470: Does getSettings() reflect configured or actual settings? (Jan-Ivar)**
    - **Issue 466: Question about setting belong to source in Section 3 (Jan-Ivar)**

CR-Blocking WebRTC-PC Issues

- Issue 1178/PR 1623: Need to describe when ICE and DTLS transport objects are created/changed (Taylor)
- Issue 1406/PR 1631: When ICE restart results in connection to a new endpoint (Taylor)
- Issue 1283/PR 1570: Centering, Scaling, Cropping (Stefan)

Issue 1178/Issue 1406: Defining the scope of DTLS/ICE transport objects

- Necessary decisions were already made at the last virtual interim.
- PR 1631 adds some prose to clarify the scope RTCDtlsTransports and RTCIceTransports.
- PR 1623 specifies when the objects are created and set on senders/receivers, as a result of setLocalDescription or setRemoteDescription.

Issue 1178/Issue 1406: Defining the scope of DTLS/ICE transport objects (cont)

PR 1631:

Each `RTCIceTransport` object represents the ICE transport layer for the RTP or RTCP `component` of a specific `RTCRtpTransceiver` , or a group of `RTCRtpTransceiver` s if such a group has been negotiated via [BUNDLE].

> NOTE
>
> An ICE restart for an existing `RTCRtpTransceiver` will be represented by an existing `RTCIceTransport`object, whose `state` will be updated accordingly, as opposed to being represented by a new object.

...

Each `RTCDtlsTransport` object represents the DTLS transport layer for the RTP or RTCP `component` of a specific `RTCRtpTransceiver` , or a group of `RTCRtpTransceiver` s if such a group has been negotiated via [BUNDLE].

> NOTE
>
> A new DTLS association for an existing `RTCRtpTransceiver` will be represented by an existing`RTCDtlsTransport` object, whose `state` will be updated accordingly, as opposed to being represented by a new object.

Issue 1178/Issue 1406: Defining the scope of DTLS/ICE transport objects (cont)

Excerpt from PR 1623:

5. If the media description is indicated as using an existing media transport according to [BUNDLE], let *transport* and *rtcpTransport* be the `RTCDtlsTransport` objects representing the RTP and RTCP components of that transport, respectively.
6. Otherwise, let *transport* and *rtcpTransport* be newly created `RTCDtlsTransport` objects, each with a new underlying `RTCIceTransport` . Though if RTCP multiplexing is negotiated according to [RFC5761], or if connection's `RTCRtcpMuxPolicy` is `require` , do not create any RTCP-specific transport objects, and instead let *rtcpTransport* equal *transport*.
7. Set transceiver.[[Sender]].[[SenderTransport]] to *transport*.
8. Set transceiver.[[Sender]].[[SenderRtcpTransport]] to *rtcpTransport*.
9. Set transceiver.[[Receiver]].[[ReceiverTransport]] to *transport*.
10. Set transceiver.[[Receiver]].[[ReceiverRtcpTransport]] to *rtcpTransport*.

Issue 1283/PR 1570: Centering, Scaling, Cropping (Stefan)

- WebRTC-PC Section 5.2:

When sending media, the sender may need to rescale or resample the media to meet various requirements including the envelope negotiated by SDP. When resizing video, the source video is first centered relative to the desired video then scaled down the minimum amount such that the video fully covers the desired size, then finally cropped to the destination size. The video remains centered while scaling and cropping. For example, if the source video was 1280 by 720, and the max size that could be sent was 640 by 480, the video would be scaled down by 1.5 and 160 columns of pixels on both the right and left sides of the source video would be cropped off. This algorithm is designed to minimize occurrence of images with with letter box or or pillow boxing. The media must not be upscaled to create fake data that did not occur in the input source.

Issue 1283/PR 1570: Centering, Scaling, Cropping (Stefan)

- WebRTC-PC Section 5.2:

... When resizing video, the source video is first centered relative to the desired video then scaled down the minimum amount such that the video fully covers the desired size, then finally cropped to the destination size. The video remains centered while scaling and cropping. ...

- “Desired” and “destination” is undefined
- It can be guessed that if the video is displayed in a video element on the receiving end, its dimensions would represent the “desired” size
- However, the only things the sender can know are:
    - Sender/Encoder capabilities
    - Receiver/Decoder capabilities (via “imageattr”)
- Video element dimensions are _not_ signaled between receiver and sender.

Issue 1283/PR 1570: Centering, Scaling, Cropping (Stefan)

- WebRTC-PC Section 5.2:

... When resizing video, the source video is first centered relative to the desired video then scaled down the minimum amount such that the video fully covers the desired size, then finally cropped to the destination size. The video remains centered while scaling and cropping. … For example, if the source video was 1280 by 720, and the max size that could be sent ...

- Possible unintended effects if “desired” and “destination” are interpreted (as they are in the example above) to be the intersection of sender and receiver capabilities
  - Say a 1920 by 1080 (16:9) track has to be resized for transmission, and the max size that be sent is 1440 by 1080 (4:3), and that the rendering video element has dimensions 960 by 540 (16:9)
  - The webrtc-pc algo says that we would discard perfectly usable data (240 pixel columns on each side), and the (default) result would be pillarboxing at the rendering video element
  - With JSEP we would (“should”) scale 1920*1080 to 1440*810, encode&send, and then scale further to 960*540 when rendering - no data thrown away

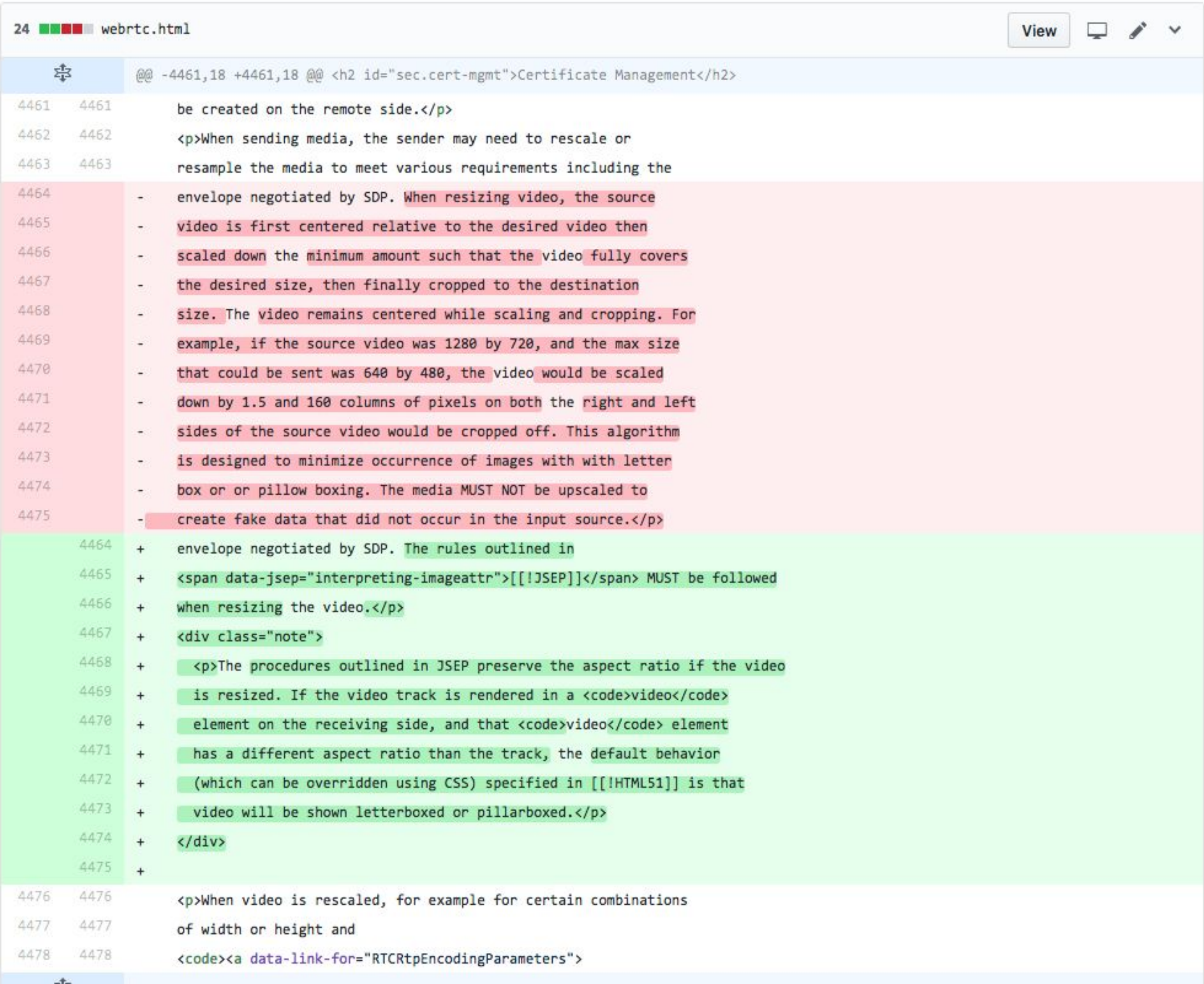
Issue 1283: Continued

- JSEP says:

If the original resolution exceeds the size limits in the attribute, the sender SHOULD apply downscaling to the output of the MediaStreamTrack in order to satisfy the limits. Downscaling MUST NOT change the track aspect ratio.

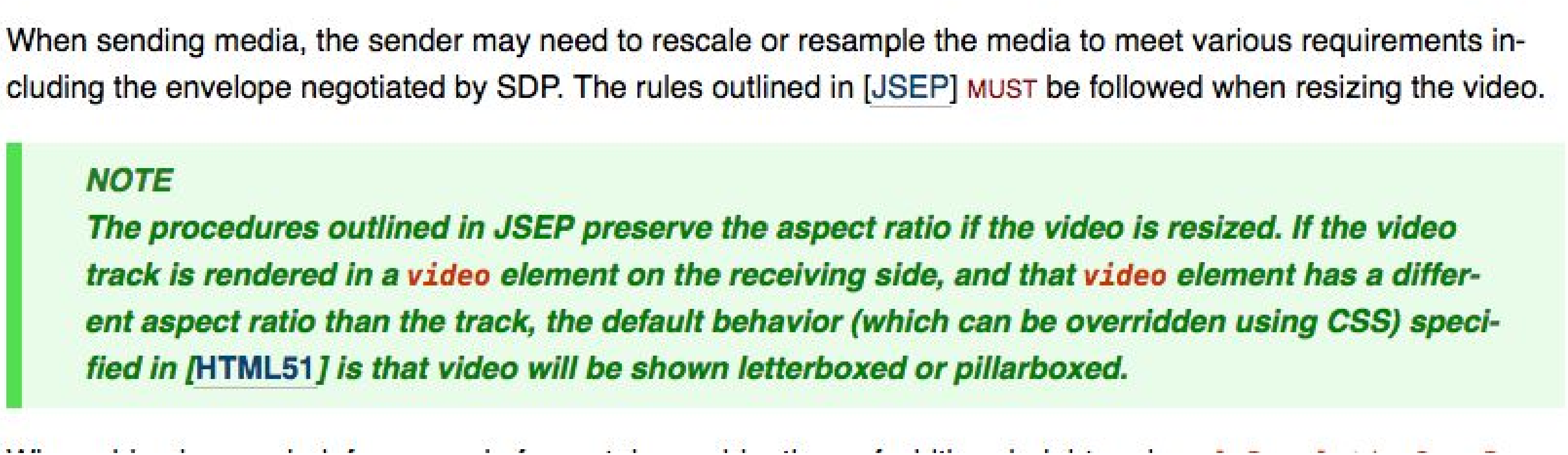
- PR 1570 proposes that webrtc-pc text on “center, scale, crop” is replaced by a reference to JSEP.
    - Also, a note explaining what happens if there is an aspect ratio mismatch between the track and a video element used for rendering (a note since this is specified in HTML and CSS)

PR 1570: Proposed resolution: Align with JSEP



```
@@ -4461,18 +4461,18 @@ <h2 id="sec.cert-mgmt">Certificate Management</h2>
     be created on the remote side.</p>
     <p>When sending media, the sender may need to rescale or
     resample the media to meet various requirements including the
-    envelope negotiated by SDP. When resizing video, the source
-    video is first centered relative to the desired video then
-    scaled down the minimum amount such that the video fully covers
-    the desired size, then finally cropped to the destination
-    size. The video remains centered while scaling and cropping. For
-    example, if the source video was 1280 by 720, and the max size
-    that could be sent was 640 by 480, the video would be scaled
-    down by 1.5 and 160 columns of pixels on both the right and left
-    sides of the source video would be cropped off. This algorithm
-    is designed to minimize occurrence of images with with letter
-    box or or pillow boxing. The media MUST NOT be upscaled to
-    create fake data that did not occur in the input source.</p>
+    envelope negotiated by SDP. The rules outlined in
+    <span data-jsep="interpreting-imageattr">[[!JSEP]]</span> MUST be followed
+    when resizing the video.</p>
+    <div class="note">
+      <p>The procedures outlined in JSEP preserve the aspect ratio if the video
+      is resized. If the video track is rendered in a <code>video</code>
+      element on the receiving side, and that <code>video</code> element
+      has a different aspect ratio than the track, the default behavior
+      (which can be overridden using CSS) specified in [[!HTML51]] is that
+      video will be shown letterboxed or pillarboxed.</p>
+    </div>
+
     <p>When video is rescaled, for example for certain combinations
     of width or height and
     <code><a data-link-for="RTCRtpEncodingParameters">
```

PR 1570: Proposed resolution: Align with JSEP



When sending media, the sender may need to rescale or resample the media to meet various requirements including the envelope negotiated by SDP. The rules outlined in [JSEP] MUST be followed when resizing the video.

> ***NOTE***
>
> *The procedures outlined in JSEP preserve the aspect ratio if the video is resized. If the video track is rendered in a `video` element on the receiving side, and that `video` element has a different aspect ratio than the track, the default behavior (which can be overridden using CSS) specified in [HTML51] is that video will be shown letterboxed or pillarboxed.*

Hoping we got agreement to merge the PRs discussed

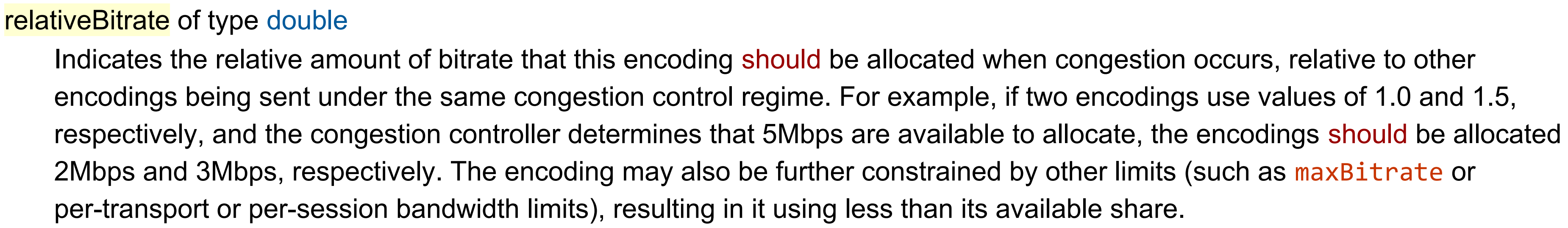
Time to request transition to CR!!!

Non-CR-blocking WebRTC-PC Issues

- Issue 1625/PR 1632: RTCPriorityType undesirably combines relative bitrate with QoS priority (Taylor)

Issue 1625/PR 1632: RTCPriorityType undesirably combines relative bitrate with QoS priority

- RTCPriorityType has some problems:
  - It controls relative “transmission capacity” of encodings, but only supports ratios of 1:2:4:8, which are not granular enough to be very useful.
  - It mixes up relative bitrate with QoS priority. There’s no way for an encoding to be given fewer bits with a higher QoS priority, which is pretty common.
- PR 1632 proposes this:



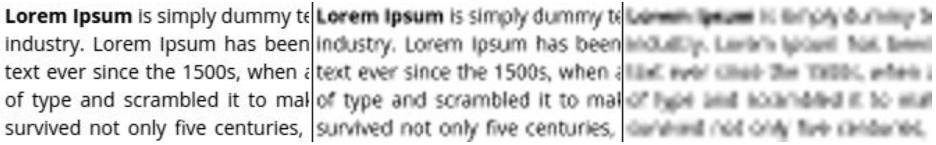
relativeBitrate of type double

Indicates the relative amount of bitrate that this encoding should be allocated when congestion occurs, relative to other encodings being sent under the same congestion control regime. For example, if two encodings use values of 1.0 and 1.5, respectively, and the congestion controller determines that 5Mbps are available to allocate, the encodings should be allocated 2Mbps and 3Mbps, respectively. The encoding may also be further constrained by other limits (such as `maxBitrate` or per-transport or per-session bandwidth limits), resulting in it using less than its available share.

Media Capture

- Issue 478: Content hints for MediaStreamTrack (Peter Boström)
- Issue 470: Does getSettings() reflect configured or actual settings? (Jan-Ivar)
- Issue 466: Question about setting belong to source in Section 3 (Jan-Ivar)

Issue 478: Content hints for MediaStreamTrack (Peter Boström)



Lorem Ipsum is simply dummy te
industry. Lorem Ipsum has been
text ever since the 1500s, when a
of type and scrambled it to mak
survived not only five centuries,

Lorem Ipsum is simply dummy te
industry. Lorem Ipsum has been
text ever since the 1500s, when a
of type and scrambled it to mak
survived not only five centuries,

- Browsers configure implicit settings based on content source.

  Chrome/WebRTC: UVC -> webcam, tab/desktop capture -> screenshare, all audio -> speech

  Wrong for capture cards, **wrong** behavior when screensharing video / game content, **wrong** for music.

- Use a content “hint” to help the browser make implicit decisions.

  Using a MST property -> can be used by MediaStreamRecorder and other APIs outside WebRTC with less flexible controls, without having to modify their specifications (scales better). Informs “balanced”.

- Example behavior:

  Motion video: Downscale / use higher max QP to preserve motion.

  Detail video: Drop frames / use lower max QPs to preserve individual frame quality.

  Speech: Use noise suppression and echo cancellation by default. Maybe enhance intelligibility?

  Music: Turn off noise suppression (preserve snares), tune echo cancellation differently / turn it off.

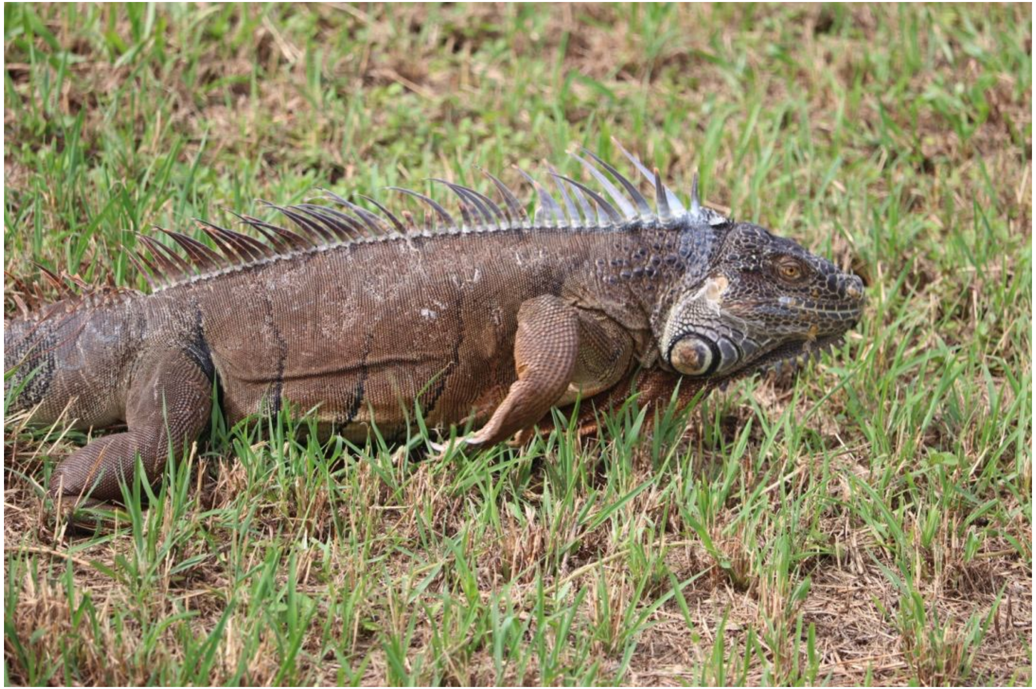
Issue 470: Does getSettings() reflect configured or actual settings? (Jan-Ivar)

- Is it a settings arbitration API (between concurrent users) or a measurement API?
- Spec says *"To check which ConstraintSets are currently in effect, the application should use getSettings."*, suggesting deterministic target “settings” values.
- Live measured values sometimes deviate from their (target) “settings”, like during:
  - camera motor pan 30 → 60: actual < setting, actual > old {pan: {max: 30}}
  - System overload or low light: measured frameRate fluctuates below target.
  - Live volume vs volume setting.
- If getSettings() were to return live values, then the spec text above doesn’t hold.
- Does any browser implement actual values? (getSettings() in Firefox returns setting).
- Does any browser implement aggressive OverconstrainedError or `onoverconstrained`? No, cause auto-disabling != useful, would surprise users at this point who might use `exact` to force rescaling/decimating. Rare users who want this behavior are better off JS measuring.
- Must agree when promise resolves, or suffer users polling getSettings() waiting on motor.

Issue 466: Question about setting belong to source in Section 3 (Jan-Ivar)

- (Note: Github discussion devolved into discussing previous slide and `ideal`)
- @guido and I are on the same page on core question above:
  - Sources practically support more than one setting concurrently. True for audio filters.
  - `track.getSettings()` should return what's relevant to consumers of that track.
  - Which means getSettings() from different tracks from the same source may return different values due to downscaling/decimating/audio processing.
  - Tracks are useful abstraction APIs for browsers. Actual hardware source settings seem of little relevance, and mandating their examination is a cross-origin security issue (can be used to detect concurrent use, e.g. is user using a particular site atm, or even be used to morse-code data across origins, bypassing cross-origin protections).
- Suggestion: Massage language as needed to reflect this. If this means settings "belong" to tracks, so be it, but if there are other ways to refine it, swell.

For extra credit



Name that lizard!

Thank you

Special thanks to:

W3C/MIT for WebEx

WG Participants, Editors & Chairs

The lizard (hope it survived Irma)