W3C WebRTC WG Meeting

April 26, 2022
8 AM - 10 AM

Chairs: Bernard Aboba
Harald Alvestrand
Jan-Ivar Bruaroey

W3C WG IPR Policy

- This group abides by the W3C Patent Policy https://www.w3.org/Consortium/Patent-Policy/
- 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 April 2022 interim meeting of the W3C WebRTC WG, at which we will cover:
    - WebNN/mediacapture-transform integration
    - WebRTC-Extensions & WebRTC-PC
    - Voice Isolation
    - Suggested Content Hint
    - Avoid user confusion by avoiding offering undesired audio sources
    - Region Capture
- Future meetings:
    - May 17
    - June 21
    - July 19

About this Virtual Meeting

- Meeting info:
  - https://www.w3.org/2011/04/webrtc/wiki/April 26_2022
- Link to latest drafts:
  - https://w3c.github.io/mediacapture-main/
  - https://w3c.github.io/mediacapture-extensions/
  - https://w3c.github.io/mediacapture-image/
  - https://w3c.github.io/mediacapture-output/
  - https://w3c.github.io/mediacapture-screen-share/
  - https://w3c.github.io/mediacapture-record/
  - https://w3c.github.io/mediacapture-viewport/
  - https://github.com/w3c/mediacapture-transform
  - https://github.com/w3c/mediacapture-handle/
  - https://w3c.github.io/webrtc-pc/
  - https://w3c.github.io/webrtc-extensions/
  - https://w3c.github.io/webrtc-stats/
  - https://w3c.github.io/mst-content-hint/
  - https://w3c.github.io/webrtc-priority/
  - https://w3c.github.io/webrtc-nv-use-cases/
  - https://github.com/w3c/webrtc-encoded-transform
  - https://github.com/w3c/webrtc-svc
  - https://github.com/w3c/webrtc-ice
- Link to Slides has been published on WG wiki
- Scribe? IRC http://irc.w3.org/ Channel: #webrtc
- The meeting is (still) being recorded. The recording will be public.
- Volunteers for note taking?

W3C Code of Conduct

- This meeting operates under W3C Code of Ethics and Professional Conduct

- We're all passionate about improving WebRTC and the Web, but let's all keep the conversations cordial and professional

Virtual Interim Meeting Tips

This session is (still) being recorded

- **Type +q and -q in the Google Meet chat to get into and out of the speaker queue.**
- **Please use headphones when speaking to avoid echo.**
- **Please wait for microphone access to be granted before speaking.**
- **Please state your full name before speaking.**
- **Poll mechanism may be used to gauge the "sense of the room".**

Understanding Document Status

- Hosting within the W3C repo does ***not*** imply adoption by the WG.
  - WG adoption requires a Call for Adoption (CfA) on the mailing list.
- Editor’s drafts do ***not*** represent WG consensus.
  - WG drafts ***do*** imply consensus, once they’re confirmed by a Call for Consensus (CfC) on the mailing list.
  - Possible to merge PRs that may lack consensus, if a note is attached indicating controversy.

Issues for Discussion Today

- 08:10 - 08:30 AM WebNN/mediacapture-transform integration
  - 08:10 - 08:20 Slides
  - 08:20 - 08:30 Discussion
- 08:30 - 08:50 AM WebRTC-Extensions & WebRTC-PC (Bernard)
  - Slides (08:30 - 08:40)
  - Discussion (08:40 - 08:50)
- 08:50 - 09:10 AM Voice Isolation Constraints (Harald)
  - Slides (08:50 - 09:00)
  - Discussion (09:00 - 09:10)
- 09:10 - 09:25 AM Suggested Content Hint (Elad)
  - Slides (09:10 - 09:15)
  - Discussion (09:15 - 09:25)
- 09:25 - 09:40 AM Avoid user-confusion by avoiding offering undesired audio sources (Elad)
  - Slides (09:25 - 09:30)
  - Discussion (09:30 - 09:40)
- 09:40 - 9:55 Region Capture (Youenn)
  - Slides (09:40 - 09:48)
  - Discussion (09:48 - 9:55)
- 9:55 - 10:00 AM Wrap-up and Next Steps

Time control:

- A warning will be given 2 minutes before time is up.
- Once time has elapsed we will move on to the next item.

WebNN/mediacapture-transform integration

Start Time: 8:10 AM

End Time: 8:30 AM

webnn/issues/226: **Integration with real-time video processing**

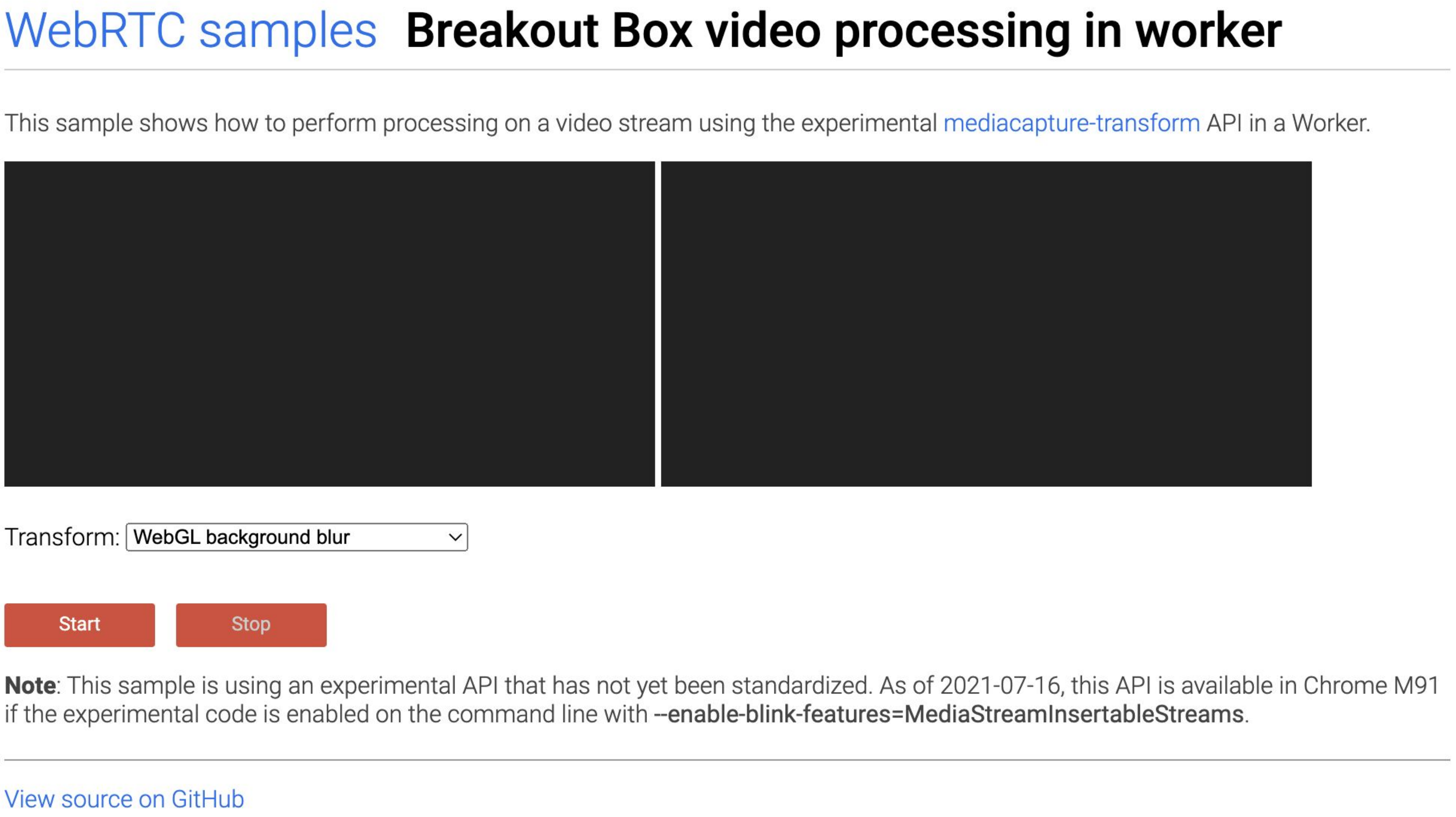
- Opened on November 10, 2021.
- Suggested goals:
    - Build a prototype that integrates the mediacapture-transform API (in a worker context) with a TF.js model that allows for background blur.
    - Measure performance of the prototype across various TF.js backends, including a WebNN-native one.
        - Ideally this would include specific measurements of memory copies, although the raw result on FPS may already give sufficient hints.
    - Focus on video processing.
        - Some of the major performance risks (in particular in terms of memory management, and potential transitions between CPU and GPU processing) will only be surfaced with video processing.

webnn/issues/226: Issues raised in the thread

- Garbage Collection
  - PR in progress: Report videoframe external memory as released on close
  - This allows the GC system to not garbage collect frequently when VideoFrames are correctly closed. Without this change GC would occur frequently because despite calling close()... there is external memory still allocated and there is an attempt to garbage collect it.
- Copy removal
  - Dom: “My reading of the code shows that there is at least a GPU→CPU transfer when turning the video frame into an input tensor; I'm not sure if the model inference is happening on the CPU or GPU. Ideally, and notwithstanding @anssiko's remarks about making this running in a worker, we would want to write a full-GPU-only pipeline, if at all possible with no memory copy. Can you already identify gaps in the APIs that would make this hard or impossible?”
  - Ningxin: “There is a corresponding import video frame to GPU texture extension/proposal: WebGL WEBGL_webcodecs_video_frame Extension and import VideoFrame from WebCodec to WebGPU proposal. So it looks like possible that the app can avoid the GPU->CPU transfer by importing the video frame into a GPU texture and feeding it into the WebNN graph which is created from the same GPU device.”
- Issue 2500: WebNN/WebGPU Integration

Video Processing in Workers

https://huningxin.github.io/webrtc-samples/src/content/insertable-streams/video-processing-worker/



WebRTC samples **Breakout Box video processing in worker**

This sample shows how to perform processing on a video stream using the experimental mediacapture-transform API in a Worker.

Transform: WebGL background blur

Note: This sample is using an experimental API that has not yet been standardized. As of 2021-07-16, this API is available in Chrome M91 if the experimental code is enabled on the command line with --**enable-blink-features=MediaStreamInsertableStreams**.

View source on GitHub

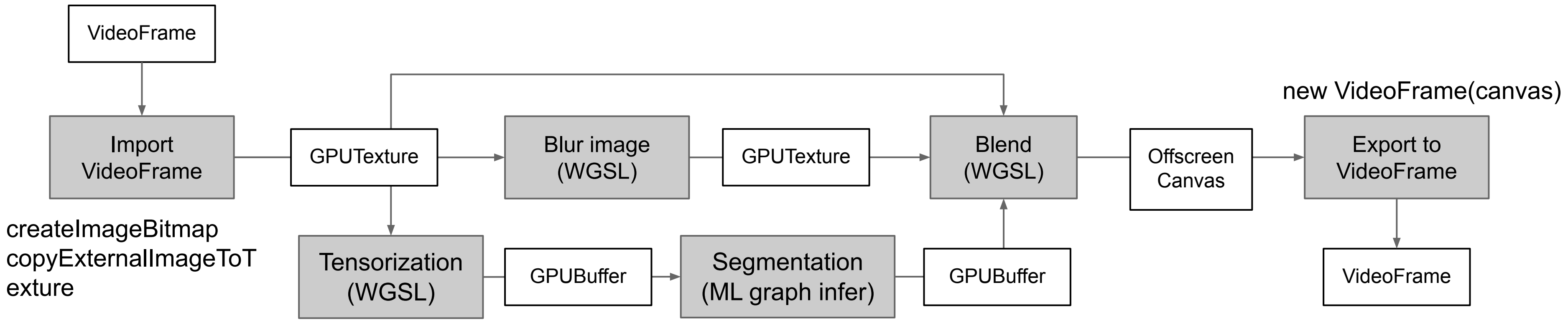
Video Processing in Workers (cont’d)

- Details of the WebGL processing pipeline (webgl-background-blur.js):
    1. VideoFrame import: VideoFrame - (createImageBitmap) -> ImageBitmap - (gl.texImage2D) -> Texture.
    2. Image blur: the shader implementation is based on @Volcomix 's virtual-background project (thanks!).
    3. Background segmentation: it is based TF.js WebGL backend that runs the TF.js DeepLabV3 model.
    4. Image blend: the segmentation result of TF.js is copied into a texture (Tensor.dataToGPU). Another WebGL frangment shader is used to blend the original input and the blurred one based on the result texture. The final output is drawn into an offscreen canvas.
    5. VideoFrame export: create VideoFrame from the offscreen canvas.

Video Processing in Workers (cont'd)

- Details of the WebGPU/WebNN processing pipeline (webgpu-background-blur.js):
  1. VideoFrame import: VideoFraome - (createImageBitmap) -> ImageBitmap - (GPUQueue::copyExternalImageToTexture) -> GPUTexture
  2. Image blur: the shader implementation is based on @austinEng 's WebGPU samples project (thanks!).
  3. Input tensor preprocessing: it is implemented in a WebGPU compute shader. Its input is a GPUTexture and the output is a GPUBuffer. The GPUBuffer will feed to WebNN graph compute as the input.
  4. Background segmentation: it is implemented by a WebNN graph (webnn_deeplabv3.js). The weights come from the TFLite DeepLabV3 model. This TFLite model and TF.js DeepLabV3 model (used by WebGL pipeline) are based on the same TF model (tensorflow/deeplabv3/1).
  5. Image blend: WebNN graph puts the segmentation results (segmap) into the output GPUBuffer. Another WebGPU compute shader is used to blend the original input and the blurred one based on the segmap. The final output is drawn into an offscreen canvas.
  6. VideoFrame export: create VideoFrame from the offscreen canvas

WebNN/mediacapture-transform integration



Opens:

- Reduce the CPU usage, current sample spends 35% on createImageBitmap and 20% on GC
  - More efficient video import?
  - More static pipeline?
- Flow control?
  - The WebGL-based processing pipeline would hang UI on entry level GPU

More details available at: [webnn/issues/226]

Discussion (End Time: 8:30 AM)

-

WebRTC-Extensions & WebRTC-PC (Bernard)

Start Time: 8:30 AM

End Time: 8:50 AM

For Discussion Today

- **WebRTC-Extensions**
  - **Issue 95: Deprecate audio/video enumeration in getCapabilities in favour of Media Capabilities API**
  - **Issue 100: Allow having inactive by default codecs**
- **WebRTC-PC Simulcast Issues**
  - **Issue 2722: sRD(offer) completely overwrites pre-existing transceiver.[[Sender]].[[SendEncodings]]**
  - **Issue 2723: The prose around "simulcast envelope" falsely implies that simulcast encodings can never be removed**
  - **Issue 2724: The language around setting a description appears to prohibit renegotiation of RIDs**

Issue 95: Deprecate audio/video enumeration in getCapabilities in favour of Media Capabilities API

- Issue 185: Retrieving RTCRtpCodecCapability from MediaCapabilities when queried with type = ‘webrtc’
  - `contentType` values represent audio and video codecs.
    - profile-id can be included in `contentType`.
  - `result` returned by `mediaCapabilities` provides `RTCRtpCodecCapability` dictionary in `result.webrtcCodec`.
  - `result.webrtcCodec` can then be used as input to `setCodecPreferences()` to select the preferred video codec for a transceiver.

Issue 185: Example

```
let mediaConfig = {`
  type: 'webrtc'.`
  audio: {
    contentType: 'audio/opus',
    channels: '2',
    bitrate: 132266,
    samplerate: 48000
  },
  video: {
    contentType: 'video/VP9; profile-id=1',
    width: 1280,
    height: 720,
    bitrate: 1234567,
    framerate: '25'
  }
};
```

```
result = {
  supported: true,
  smooth: true,
  powerEfficient: false,
  webrtcCodec: {
    clockRate: 90000,
    mimeType: 'video/VP9',
    sdpFmtpLine: 'profile-id=0'
  }
}
```

```
result = await navigator.mediaCapabilities.decodingInfo(mediaConfig)
```

Issue 95: Deprecate audio/video enumeration in getCapabilities (cont'd)

- Scope comparison:
  - `mediaCapabilities()` returns information relating to audio and video codecs supported by WebRTC.
  - `RTCRtpSender/Receiver.getCapabilities().headerExtensions` returns information on supported header extensions.
  - `RTCRtpSender/Receiver.getCapabilities().codecs` returns information on audio/video codecs as well as telephone-event, CN, FEC, RTX, RED, etc.

Issue 95: RTCRtpReceiver.getCapabilities() Audio Example https://webrtc.internaut.com/iit-2020/cap-dumper/

```
RTCRtpReceiver.getCapabilities(audio):
{
  "codecs": [
    {
      "channels": 2,
      "clockRate": 48000,
      "mimeType": "audio/red",
      "sdpFmtpLine": "111/111"
    },
    {
      "channels": 1,
      "clockRate": 32000,
      "mimeType": "audio/CN"
    },
    {
      "channels": 1,
      "clockRate": 16000,
      "mimeType": "audio/CN"
    },
    {
      "channels": 1,
      "clockRate": 48000,
      "mimeType": "audio/telephone-event"
    },
    {
      "channels": 1,
      "clockRate": 8000,
      "mimeType": "audio/telephone-event"
    }
  ],
```

Issue 95: RTCRtpReceiver.getCapabilities() Video Example https://webrtc.internaut.com/iit-2020/cap-dumper/

RTCRtpReceiver.getCapabilities(video):

```
{
 "codecs": [
   {
    "clockRate": 90000,
    "mimeType": "video/rtx"
   },
   {
    "clockRate": 90000,
    "mimeType": "video/VP9",
    "sdpFmtpLine": "profile-id=0"
   },
    {
    "clockRate": 90000,
    "mimeType": "video/red"
   },
   {
    "clockRate": 90000,
    "mimeType": "video/ulpfec"
   },
   {
    "clockRate": 90000,
    "mimeType": "video/flexfec-03",
    "sdpFmtpLine": "repair-window=10000000"
     }
```

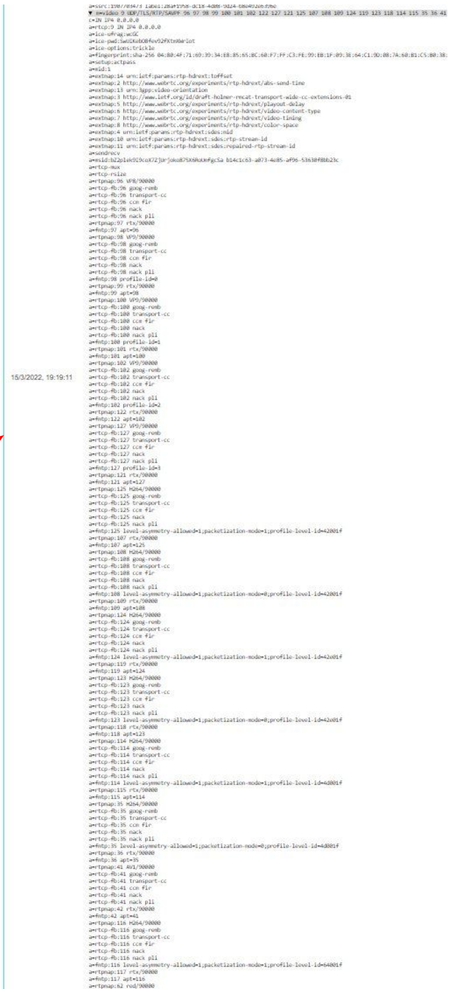
Issue 95: Questions

- Should `mediaCapabilities()` for WebRTC provide information on all the codecs covered `getCapabilities()`?
  - `telephone-event, CN, FEC, RTX, RED, etc.`
  - Would these codecs ever be `power-efficient` or `smooth?`
    - Would width, height, framerate, bitrate ever affect the result?
- Is it a goal for `mediaCapabilities()` to deprecate `getCapabilities()`?

Issue 100: Allow having inactive by default codecs

- Sergio:
  - “I am currently adding support for vp9 profiles 1 and 3 and h264 Hi10P and Hi422P to libwebrtc, and the SDP generated is HUGE (see next slide).
  - Most of these codec/profiles won't be used on a daily basis, so I would be great if there could be a way of having them disabled by default, so they don't appear on the offer (but they are negotiated on the answer) and have an API to enable them by the app if needed. I don't think we have anything similar at the moment.”
- Fippo:
  - “We've run into similar issues before, both when enabling flexfec-03 (recvonly because uhm...) as well as the (recvonly) YUV444 addition that comes to Chrome 101 and this has already resulted in running out of payload types in the upper range.... setCodecPreferences is great but we have the same problem as with Issue 99.”

Issue 100: Allow having inactive by default codecs (cont'd)



15/3/2022, 19:19:11

```
a=ssrc:1907784473 label:28af1958-dc18-4d08-9d24-6b0492d63960
▼ m=video 9 UDP/TLS/RTP/SAVPF 96 97 98 99 100 101 102 122 127 121 125 107 108 109 124 119 123 118 114 115 35 36 41
c=IN IP4 0.0.0.0
a=rtcp:9 IN IP4 0.0.0.0
a=ice-ufrag:wcGC
a=ice-pwd:SwUIKobO8fev92fXtmXWrlot
a=ice-options:trickle
a=fingerprint:sha-256 04:80:4F:71:69:39:34:E8:85:65:BC:60:F7:FF:C3:FE:99:EB:1F:09:3E:64:C1:9D:08:7A:60:B1:C5:B0:38:
a=setup:actpass
a=mid:1
a=extmap:14 urn:ietf:params:rtp-hdrext:toffset
a=extmap:2 http://www.webrtc.org/experiments/rtp-hdrext/abs-send-time
a=extmap:13 urn:3gpp:video-orientation
a=extmap:3 http://www.ietf.org/id/draft-holmer-rmcat-transport-wide-cc-extensions-01
a=extmap:5 http://www.webrtc.org/experiments/rtp-hdrext/playout-delay
a=extmap:6 http://www.webrtc.org/experiments/rtp-hdrext/video-content-type
a=extmap:7 http://www.webrtc.org/experiments/rtp-hdrext/video-timing
a=extmap:8 http://www.webrtc.org/experiments/rtp-hdrext/color-space
a=extmap:4 urn:ietf:params:rtp-hdrext:sdes:mid
a=extmap:10 urn:ietf:params:rtp-hdrext:sdes:rtp-stream-id
a=extmap:11 urn:ietf:params:rtp-hdrext:sdes:repaired-rtp-stream-id
a=sendrecv
a=msid:b22plek9I9coX72jUrjoko875X6RoUnfgcSa b14c1c63-a073-4e85-af96-53638f8bb23c
a=rtcp-mux
a=rtcp-rsize
a=rtpmap:96 VP8/90000
a=rtcp-fb:96 goog-remb
a=rtcp-fb:96 transport-cc
a=rtcp-fb:96 ccm fir
a=rtcp-fb:96 nack
a=rtcp-fb:96 nack pli
a=rtpmap:97 rtx/90000
a=fmtp:97 apt=96
a=rtpmap:98 VP9/90000
a=rtcp-fb:98 goog-remb
a=rtcp-fb:98 transport-cc
a=rtcp-fb:98 ccm fir
a=rtcp-fb:98 nack
a=rtcp-fb:98 nack pli
a=fmtp:98 profile-id=0
a=rtpmap:99 rtx/90000
a=fmtp:99 apt=98
a=rtpmap:100 VP9/90000
a=rtcp-fb:100 goog-remb
a=rtcp-fb:100 transport-cc
a=rtcp-fb:100 ccm fir
a=rtcp-fb:100 nack
a=rtcp-fb:100 nack pli
a=fmtp:100 profile-id=1
a=rtpmap:101 rtx/90000
a=fmtp:101 apt=100
a=rtpmap:102 VP9/90000
a=rtcp-fb:102 goog-remb
a=rtcp-fb:102 transport-cc
a=rtcp-fb:102 ccm fir
a=rtcp-fb:102 nack
a=rtcp-fb:102 nack pli
a=fmtp:102 profile-id=2
a=rtpmap:122 rtx/90000
a=fmtp:122 apt=102
a=rtpmap:127 VP9/90000
a=rtcp-fb:127 goog-remb
a=rtcp-fb:127 transport-cc
a=rtcp-fb:127 ccm fir
a=rtcp-fb:127 nack
a=rtcp-fb:127 nack pli
a=fmtp:127 profile-id=3
a=rtpmap:121 rtx/90000
a=fmtp:121 apt=127
a=rtpmap:125 H264/90000
a=rtcp-fb:125 goog-remb
a=rtcp-fb:125 transport-cc
a=rtcp-fb:125 ccm fir
a=rtcp-fb:125 nack
a=rtcp-fb:125 nack pli
a=fmtp:125 level-asymmetry-allowed=1;packetization-mode=1;profile-level-id=42001f
a=rtpmap:107 rtx/90000
a=fmtp:107 apt=125
a=rtpmap:108 H264/90000
a=rtcp-fb:108 goog-remb
a=rtcp-fb:108 transport-cc
a=rtcp-fb:108 ccm fir
a=rtcp-fb:108 nack
a=rtcp-fb:108 nack pli
a=fmtp:108 level-asymmetry-allowed=1;packetization-mode=0;profile-level-id=42001f
a=rtpmap:109 rtx/90000
a=fmtp:109 apt=108
a=rtpmap:124 H264/90000
a=rtcp-fb:124 goog-remb
a=rtcp-fb:124 transport-cc
a=rtcp-fb:124 ccm fir
a=rtcp-fb:124 nack
a=rtcp-fb:124 nack pli
a=fmtp:124 level-asymmetry-allowed=1;packetization-mode=1;profile-level-id=42e01f
a=rtpmap:119 rtx/90000
a=fmtp:119 apt=124
a=rtpmap:123 H264/90000
a=rtcp-fb:123 goog-remb
a=rtcp-fb:123 transport-cc
a=rtcp-fb:123 ccm fir
a=rtcp-fb:123 nack
a=rtcp-fb:123 nack pli
a=fmtp:123 level-asymmetry-allowed=1;packetization-mode=0;profile-level-id=42e01f
a=rtpmap:118 rtx/90000
a=fmtp:118 apt=123
a=rtpmap:114 H264/90000
a=rtcp-fb:114 goog-remb
a=rtcp-fb:114 transport-cc
a=rtcp-fb:114 ccm fir
a=rtcp-fb:114 nack
a=rtcp-fb:114 nack pli
a=fmtp:114 level-asymmetry-allowed=1;packetization-mode=1;profile-level-id=4d001f
a=rtpmap:115 rtx/90000
a=fmtp:115 apt=114
a=rtpmap:35 H264/90000
a=rtcp-fb:35 goog-remb
a=rtcp-fb:35 transport-cc
a=rtcp-fb:35 ccm fir
a=rtcp-fb:35 nack
a=rtcp-fb:35 nack pli
a=fmtp:35 level-asymmetry-allowed=1;packetization-mode=0;profile-level-id=4d001f
a=rtpmap:36 rtx/90000
a=fmtp:36 apt=35
a=rtpmap:41 AV1/90000
a=rtcp-fb:41 goog-remb
a=rtcp-fb:41 transport-cc
a=rtcp-fb:41 ccm fir
a=rtcp-fb:41 nack
a=rtcp-fb:41 nack pli
a=rtpmap:42 rtx/90000
a=fmtp:42 apt=41
a=rtpmap:116 H264/90000
a=rtcp-fb:116 goog-remb
a=rtcp-fb:116 transport-cc
a=rtcp-fb:116 ccm fir
a=rtcp-fb:116 nack
a=rtcp-fb:116 nack pli
a=fmtp:116 level-asymmetry-allowed=1;packetization-mode=1;profile-level-id=64001f
a=rtpmap:117 rtx/90000
a=fmtp:117 apt=116
a=rtpmap:62 red/90000
```

Empire State Building (not actual size)

Godzilla SDP!

Issue 100: Allow having inactive by default codecs (cont'd)

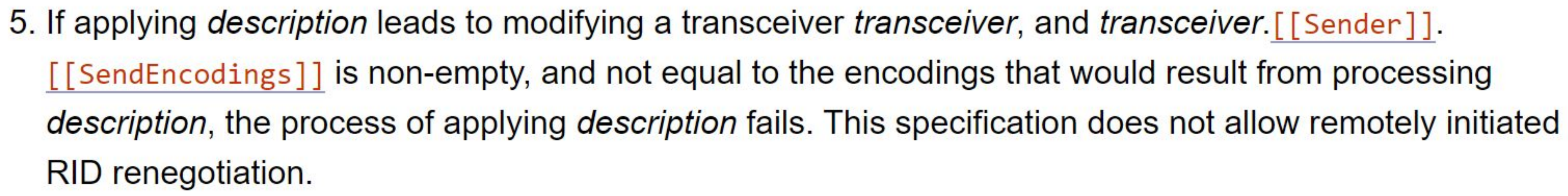
- How about adding an extension to the transceivers similar to the one for the rtp header extensions?

```
partial interface RTCRtpTransceiver {
  undefined setOfferedCodecs(sequence<RTCRtpCodecCapability> codecsToOffer);
  readonly attribute FrozenArray<RTCRtpCodecCapability> codecsToOffer;
  readonly attribute FrozenArray<RTCRtpCodecCapability> codecsNegotiated;
};
```

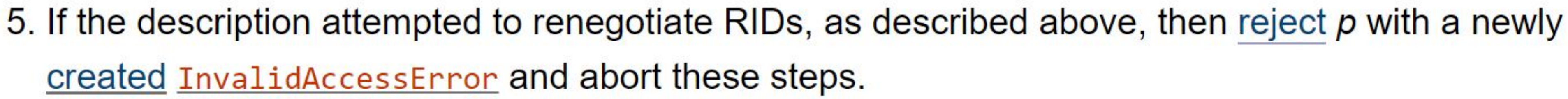
- We would need to add a property `offeredByDefault` to `RTCRtpCodecCapability` to signal if the codec is included in an initial offer by default or not.
- What does the WG think?

WebRTC-PC Simulcast Issues

- Issue 2722, Issue 2723, Issue 2724 originate from contradictions between RFC 8853 and WebRTC-PC Sections 4.4.1.5 and 5.4.1.
- Section 4.4.1.5 says:



5. If applying *description* leads to modifying a transceiver *transceiver*, and *transceiver*.`[[Sender]]`.`[[SendEncodings]]` is non-empty, and not equal to the encodings that would result from processing *description*, the process of applying *description* fails. This specification does not allow remotely initiated RID renegotiation.



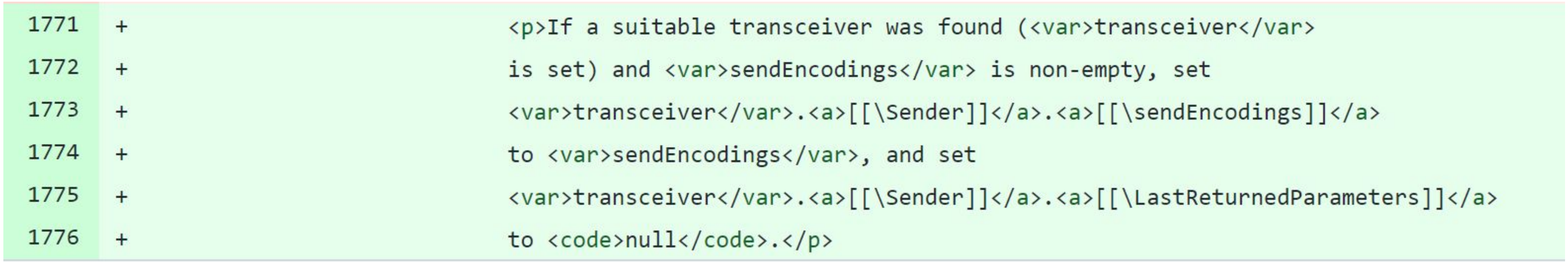
5. If the description attempted to renegotiate RIDs, as described above, then reject *p* with a newly created `InvalidAccessError` and abort these steps.

Issue 2722: sRD(offer) completely overwrites pre-existing transceiver.[[Sender]].[[SendEncodings]]

- The language that describes how to handle simulcast in a remote offer says that [[SendEncodings]] is completely replaced based on the rids in the simulcast attribute.
  - While this works fine for transceivers that are not yet associated, for already associated transceivers (which have already populated [[SendEncodings]]), this is not appropriate.
  - [BA] Over-writing is prohibited in Section 4.4.1.5.
- We need to specify what happens on sRD(offer) when there is already an associated transceiver.
  - Since we (rightly) allow sRD(answer) to remove pre-existing rids, we probably need to allow sRD(offer) to remove pre-existing rids as well (since the base simulcast spec requires the answerer to handle this situation).
  - We also need to ensure that the language around createAnswer does the right thing if the offer tries to add a rid (ie; the answer will not contain that new rid).

Issue 2722: sRD(offer) completely overwrites pre-existing transceiver.[[Sender]].[[SendEncodings]]

- PR 2155 over-writes existing transceiver:



```
+                       <p>If a suitable transceiver was found (<var>transceiver</var>
+                       is set) and <var>sendEncodings</var> is non-empty, set
+                       <var>transceiver</var>.<a>[[\Sender]]</a>.<a>[[\sendEncodings]]</a>
+                       to <var>sendEncodings</var>, and set
+                       <var>transceiver</var>.<a>[[\Sender]]</a>.<a>[[\LastReturnedParameters]]</a>
+                       to <code>null</code>.</p>
```

- Does the recommended direction make sense?
- Should we mark this Issue “Ready for PR”?

§ 5.4.1 Simulcast functionality

Simulcast functionality is provided via the `addTransceiver` method of the `RTCPeerConnection` object and the `setParameters` method of the `RTCRtpSender` object.

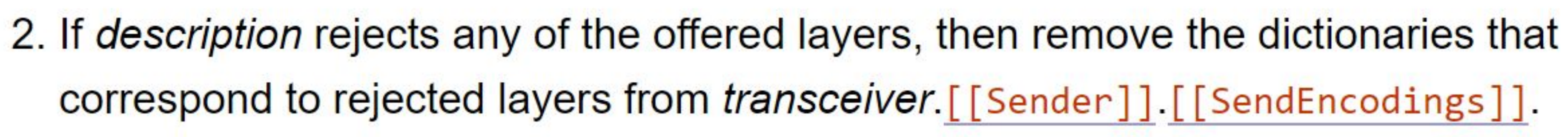
The `addTransceiver` method establishes the **simulcast envelope** which includes the maximum number of simulcast streams that can be sent, as well as the ordering of the `encodings`. While characteristics of individual simulcast streams can be modified using the `setParameters` method, the simulcast envelope cannot be changed. One of the implications of this model is that the `addTrack()` method cannot provide simulcast functionality since it does not take `sendEncodings` as an argument, and therefore cannot configure an `RTCRtpTransceiver` to send simulcast.

Another implication is that the answerer cannot set the simulcast envelope directly. Upon calling the `setRemoteDescription` method of the `RTCPeerConnection` object, the simulcast envelope is configured on the `RTCRtpTransceiver` to contain the layers described by the specified session description. Once the envelope is determined, layers cannot be removed. They can be marked as inactive by setting the `active` member to `false` effectively disabling the layer.

While `setParameters` cannot modify the simulcast envelope, it is still possible to control the number of streams that are sent and the characteristics of those streams. Using `setParameters`, simulcast streams can be made inactive by setting the `active` member to `false`, or can be reactivated by setting the `active` member to `true`. Using `setParameters`, stream characteristics can be changed by modifying attributes such as `maxBitrate`.

[Issue 2723](#): The prose around "simulcast envelope" falsely implies that simulcast encodings can never be removed (cont'd)

- Spec says "Once the envelope is determined, layers cannot be removed.", but the language for sRD(answer) says that if rids are rejected by an answer, they are removed.



2. If *description* rejects any of the offered layers, then remove the dictionaries that correspond to rejected layers from *transceiver*.`[[Sender]]`.`[[SendEncodings]]`.

[BA] This doesn't appear to be a contradiction to me, since the envelope is set via sRD(), not before.

- There are a couple of ways to fix this:
  1. We remove this assurance from the section on "simulcast envelope", or
  2. We only allow the first answer to remove rids from [[SendEncodings]].

Disallowing an answer to remove rids on a previously negotiated sender is probably not appropriate, since this would violate the simulcast spec, which requires the offerer to handle this case regardless of whether this is the initial negotiation or not. I think option 1 is the correct course of action here.

Issue 2723: The prose around "simulcast envelope" falsely implies that simulcast encodings can never be removed (cont’d)

- What does the WG want to do?
  - Does the WG believe that there is a contradiction in the spec?
  - Is there an interest in enabling re-negotiation?

Issue 2724: The language around setting a description appears to prohibit renegotiation of RIDs

- Section 4.4.1.5:
  - "5. If the description attempted to renegotiate RIDs, as described above, then reject p with a newly created `InvalidAccessError` and abort these steps."
- This prohibits a local re-offer from adding or removing RIDs.
- However, RFC 8853 indicates that an offerer cannot refuse to honor a remote answer that rejects a previously negotiated RID.
  - RFC 8853 Section 5.3.2:
    - "An answerer that receives an offer with simulcast that lists a number of simulcast streams MAY reduce the number of simulcast streams in the answer, but it MUST NOT add simulcast streams."
  - RFC 8853 Section 5.3.4:
    - "Offers inside an existing session follow the same rules as for initial SDP offer, with these additions:"

Issue 2724: The language around setting a description appears to prohibit renegotiation of RIDs (cont'd)

- RFC 8853 also indicates that an answerer can't refuse to honor a remote offer because it removed a previously negotiated RID.
    - RFC 8853 Section 5.3.3:
        - "An offerer that receives an answer where some rid-id alternatives are kept MUST be prepared to receive any of the kept "send"-directionrid-id alternatives and MAY send any of the kept "receive"-direction rid-id alternatives.
        - An offerer that receives an answer where some of the rid-ids are removed compared to the offer MAY release the corresponding resources (codec, transport, etc) in its "receive" direction and MUST NOT send any RTP packets corresponding to the removed rid-ids."
    - RFC 8853 Section 5.3.4:
        - "Creation of SDP answers and processing of SDP answers inside an existing session follow the same rules as described above for initial SDP offer/answer."

Issue 2724: The language around setting a description appears to prohibit renegotiation of RIDs (cont’d)

- What does the WG think?

RFC 8853 “Using Simulcast in SDP and RTP Sessions”

- Section 4 Overview

```
a=simulcast:send 1;2,3 recv 4
```

- If this line is included in an SDP offer, the "send" part indicates the offerer's capability and proposal to send two simulcast RTP streams.
- Each simulcast stream is described by one or more RTP stream identifiers (rid-ids), and each group of rid-ids for a simulcast stream is separated by a semicolon (";").
- When a simulcast stream has multiple rid-ids that are separated by a comma (","), they describe ***alternative representations for that particular simulcast RTP stream***. Thus, the "send" part shown above is interpreted as an intention to send two simulcast RTP streams. The first simulcast RTP stream is identified and restricted according to rid-id 1.
- The second simulcast RTP stream can be sent as ***two alternatives***, identified and restricted according to rid-ids 2 and 3.
- The "recv" part of the line shown here indicates that the offerer desires to receive a single RTP stream (no simulcast) according to rid-id 4.

RFC 8853 “Using Simulcast in SDP and RTP Sessions”

- Section 5.3.2 Creating the SDP Answer
  - An answerer that receives an offer with simulcast containing an "a=simulcast" attribute listing ***alternative rid-ids*** MAY keep all the alternative rid-ids in the answer, but it MAY also choose to remove any nondesirable ***alternative rid-ids*** in the answer.
  - The answerer MUST NOT add any ***alternative rid-ids*** in the "send" direction in the answer that were not present in the offer receive direction. The answerer MUST be prepared to receive any of the receive-direction rid-id alternatives and MAY send any of the "send"-direction alternatives that are part of the answer.
  - An answerer that receives an offer with simulcast that lists a number of simulcast streams MAY reduce the number of simulcast streams in the answer, but it MUST NOT add simulcast streams.
- Section 5.3.3 Offerer processing the SDP Answer
  - An offerer that receives an answer where some ***rid-id alternatives*** are kept MUST be prepared to receive any of the kept "send"-direction ***rid-id alternatives*** and MAY send any of the kept "receive"-direction rid-id alternatives.
  - An offerer that receives an answer where some of the rid-ids are removed compared to the offer MAY release the corresponding resources (codec, transport, etc) in its "receive" direction and MUST NOT send any RTP packets corresponding to the removed rid-ids.
- RFC 8853 does not prohibit an answer from changing the order of the rids.
- RFC 8853 does not prohibit a re-offer from changing the order of the rids.

Discussion (End Time: 8:50 AM)

-

Voice Isolation Constraints (Harald)

Start Time: 8:50 AM

End Time: 9:10 AM

Issue 62: Voice Isolation Constraint

Microphone signals contain lots of information

- Human voices
- Music
- Keyboard typing noises
- Traffic noises

Sometimes the app knows what the recipient wants to hear

A common scenario is a human voice.

Voice Isolation Constraint - How

There are multiple algorithms for isolating voice

- Stuff that works on pure audio samples
- Stuff that works with hardware support
- Stuff that invokes ML models

How it is done is not so important. The desirable outcome is important.

Voice Isolation Constraint - What

Introduce a new boolean constraint:
voiceIsolation

Define the result as “attempt to remove everything that is not a human voice”

May also choose to enhance the “most important speaker” if multiple voices

Voice Isolation Constraint - Issues

- Feature detection
  - By MediaTrackCapabilities
- Interaction with noiseSuppression
  - Probably noise=false, voice=true makes no sense
  - Ignore noiseSuppression when voiceIsolation=true

Next steps

PR for adding the constraint

- If WG indicates that it’s happy.

Discussion (End Time: 9:10 AM)

-

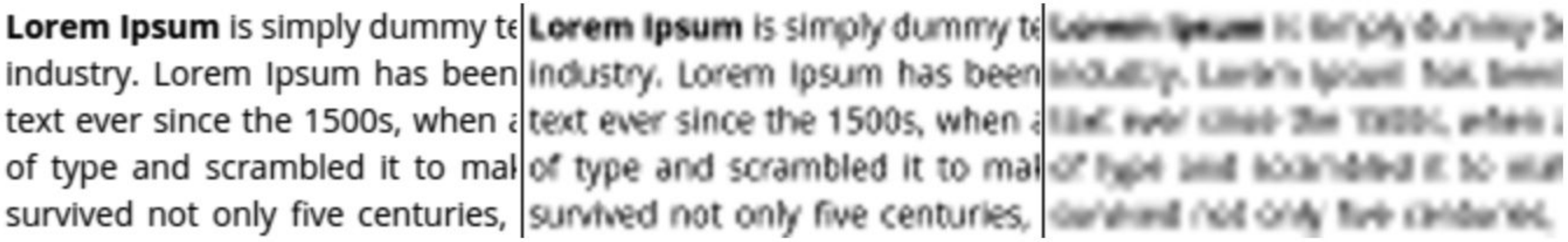
mediacapture-handle/35

Suggested Content Hint (Elad)

Start time: 9:10 AM
End time: 9:25 AM

Reminder #1: contentHint

- One can’t always transmit video at both high frame rate **and** high resolution.
- Depending on the transmitted content, it might be preferable to degrade the frame rate or the resolution.
    - With static content (e.g. slides), it is usually preferable to degrade framerate.
    - With dynamic content (e.g. video), it is often preferable to degrade resolution.
- A mechanism exists for specifying the application’s preference - contentHint.
- For video, possible values include “motion”, “detail” and “text”. Predictably:
    - contentHint = “text” results in frame rate being degraded before resolution.
    - contentHint = “motion” results in resolution being degraded before frame rate.


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

Reminder #2: Capture Handle Identity

- Capture Handle Identity is a mechanism by which a screen-captured tab reveals its identity to the capturing application.
  - Without knowing it is being captured, the captured application sets its "identity" - just in case it ends up being captured.
  - The capturing application can read this value.

Tying the two mechanisms together

What if an application could **suggest** to would-be capturers the best contentHint to apply? An application could declare “if I am captured and transmitted remotely, it’s preferable to degrade X first.”

Slide with loads of text.
`suggestedContentHint = “text”`

Slide with an embedded video.
`suggestedContentHint = “motion”`

Capturing tab

```
if (Predicate(captureHandle.origin)) {
  track.contentHint =
    captureHandle.suggestContentHint;
}
```

Where Predicate() is some function. It could examine an allowlist or a blocklist, or it could trust any captured application’s hint. It’s up to the capturing application.

Corollary - Multiple Calls to setCaptureHandleConfig()

This serves as additional motivation for allowing setCaptureHandleConfig() to be called multiple times. The content on the captured document can change - and then, so does the suggestion, conveyed over Capture Handle.

(Recall that when the CaptureHandleConfig changes in the captured tab, an event is fired in the capturing tab.)

Discussion (End Time: 9:25 AM)

-

mediacapture-screen-share/219

Avoid user-confusion by avoiding offering undesired audio sources (Elad)

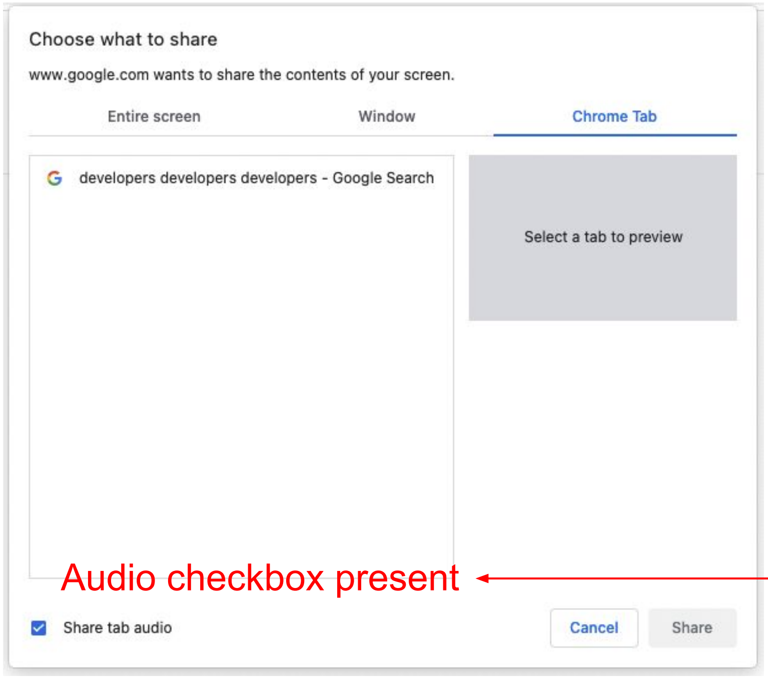
Time time: 9:25 AM
End time: 9:40 AM

Problem Statement

- Consider a video conferencing application. It plays audio from remote participants over local speakers.
- Assume the local user chooses to share their tab/screen along with tab/system audio.
- Our hypothetical VC client:
    - <u>Does</u> intend to relay captured tab-audio.
    - Does <u>not</u> intend to relay captured system-audio.
- When calling getDisplayMedia({video: …, audio: …}), it is not currently possible to specify interest in X-audio and disinterest in Y-audio.
- The user agent therefore offers any kind of audio it can support. (Because <u>other</u> applications do wish for both X- and Y-audio.)
- Users who try to share system-audio are confused that they were asked for system-audio, approved system-audio, but are not relaying remotely system-audio.

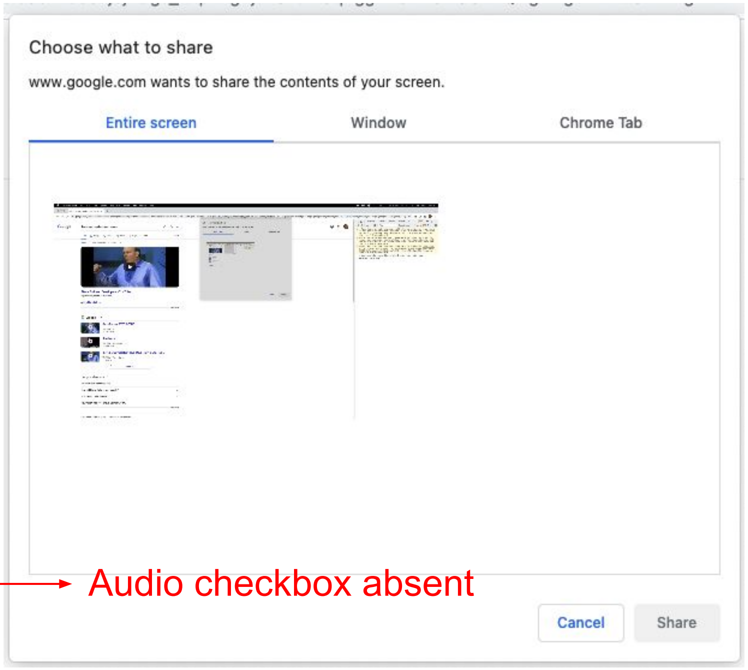
Solution - Generally Speaking

- The application needs a way to signal to the user agent, that only X-audio is requested.



Choose what to share

www.google.com wants to share the contents of your screen.



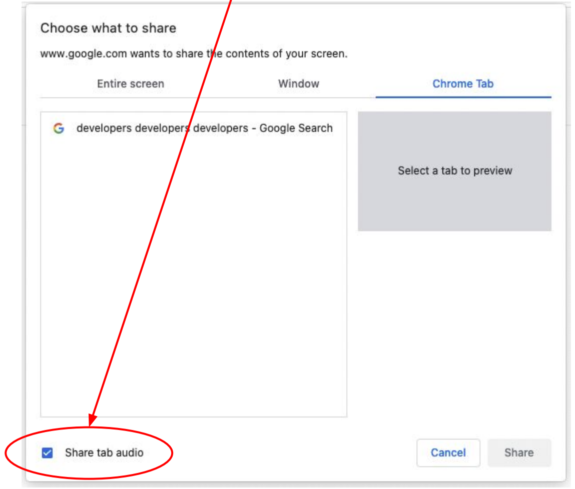
Choose what to share

www.google.com wants to share the contents of your screen.

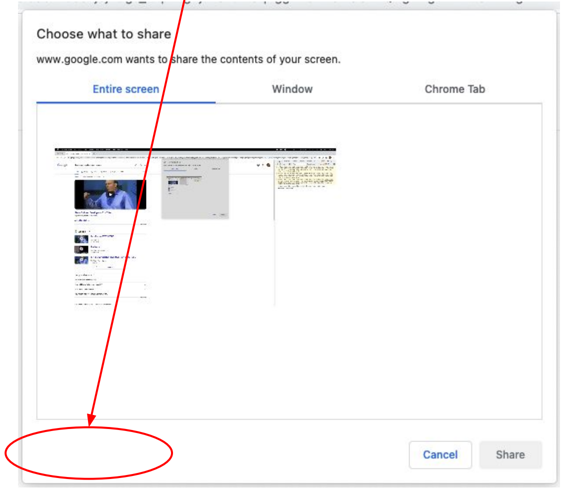
Audio checkbox present ⟷ Audio checkbox absent

Solution - Specific Suggestion

```
navigator.mediaDevices.getDisplayMedia({
   video: true,
   audio: {browserAudioDesired, ~~monitorAudioDesired~~},
});
```


Choose what to share

www.google.com wants to share the contents of your screen.


Choose what to share

www.google.com wants to share the contents of your screen.

Discussion (End Time: 9:40 AM)

-

Region Capture (Youenn)

Start Time: 9:40 AM

End Time: 9:55 AM

Issue 11: CropTarget generation API

- Various envisioned API-shape options
  - Expose API to Element or MediaDevices
  - Expose API as a method or as an attribute
  - Return result as a promise or as a CropTarget directly

Issue 11: CropTarget generation API

- Attach API to either Element or MediaDevices

```
partial interface Element {
  Promise<CropTarget> getCropTarget();
};
partial interface MediaDevices {
  Promise<CropTarget> produceCropTarget(Element
element);
};
```

- Advantages for MediaDevices option
  - Grouping capture related APIs in a media-related place
    - Easing documentation and searchability
- Advantages for Element option
  - Partial interfaces is a well-known solution for documentation/searchability
    - E.g. Element.requestFullScreen
  - Avoid corner cases by removing unnecessary interaction with MediaDevices object
    - MediaDevices is attached to a single document
      - MediaDevices is neutered when its document is detached
      - Element might be removed/added from one document to another
    - MediaDevices is SecureContext, Element is not SecureContext

Issue 11: CropTarget generation API

- In case of exposing API at Element level, API can either be a method or an attribute

```
partial interface Element {
  // As an attribute
  readonly attribute Promise<CropTarget> cropTarget;

  // As a method
  readonly Promise<CropTarget> getCropTarget();
};
```

- Attribute API mandates a single CropTarget per Element
  - Already possible to generate several CropTargets referencing the same Element
    - CropTarget is serializable
- Slight preference for a method API
  - Slightly simpler/more efficient for implementors

Issue 11: CropTarget generation API

- Using promise API or not using promise API

```
partial interface Element {
  CropTarget getCropTarget();
  Promise<CropTarget> getCropTarget();
};
partial interface MediaDevices {
  CropTarget produceCropTarget(Element element);
  Promise<CropTarget> produceCropTarget(Element element);
};
```

- Advantages for promise-based API
  - Compatible with Chrome current prototype implementation
- Advantages for non-promise-based API
  - Consistent with existing web API design
  - Easier for web developers
    - No need for async functions
    - No corner cases where promise might get rejected
      - Element moved to another document
      - Element owned by a detached element or detached document
  - OK for implementors
    - Algorithms using CropTarget (cropTo e.g.) are asynchronous

Discussion (End Time: 9:55 AM)

-

CaptureController (Jan-Ivar)
If Time Permits

CaptureController

The number of features being considered added to screen capture is growing:

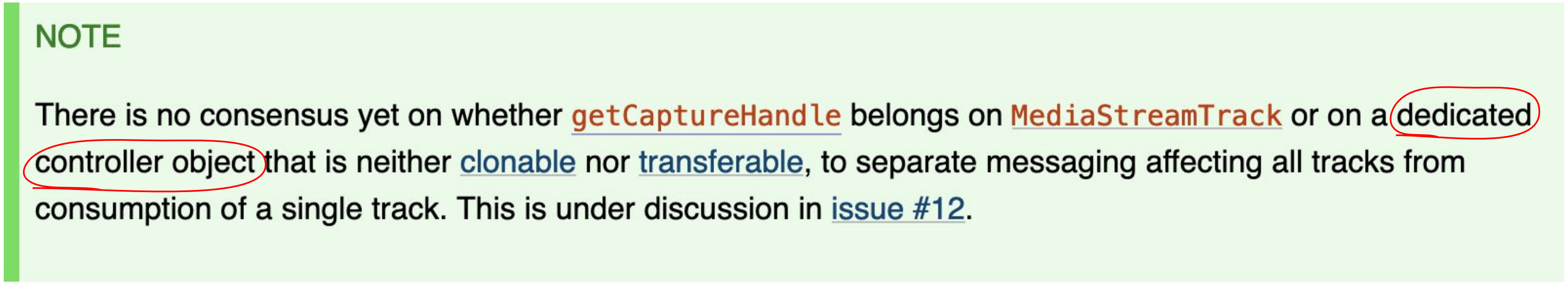
1. Capture Handle Identity
2. Capture Handle Actions
3. Conditional Focus #190
4. Multi-capture (picker multi-select)

The current getDisplayMedia API seems strained by all these additions.

A dedicated controller API might be desirable. But how to retrofit?

#12 …seems misplaced on the track

The mediacapture-handle spec notes:



NOTE

There is no consensus yet on whether `getCaptureHandle` belongs on `MediaStreamTrack` or on a dedicated controller object that is neither clonable nor transferable, to separate messaging affecting all tracks from consumption of a single track. This is under discussion in issue #12.

Tracks can be cloned and transferred to workers. Giving a cross-origin messaging channel that defeats storage partitioning, to all media consumers, seems undesirable.

Other track (consumer) settings don’t affect clones, or are arbitrated through constraints.

CaptureController

The dedicated controller object:

```
const controller = new CaptureController();
const stream = await navigator.mediaDevices.getDisplayMedia({controller});
const {origin, handle} = controller; // identity

await controller.focus(); // conditional focus #190

const actions = controller.getSupportedActions(); // actions
if (actions.includes("nextslide")) {
  await controller.sendAction("nextslide");
}
```

Or even:

```
const controller = new CaptureController();
const stream = await controller.getDisplayMedia();
```

Discussion

-

Thank you

Special thanks to:

WG Participants, Editors & Chairs