W3C WebRTC WG Meeting

May 17, 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 May 2022 interim meeting of the W3C WebRTC WG, at which we will cover:
  - WebRTC-Extensions
  - WebRTC-PC
  - CaptureController
  - WebRTC-encoded-transform
  - MediaCapture-Extensions
  - Dynamic Sources while screen-sharing

Future meetings

- Proposal to move June 28 meeting to June 7
- Next meeting after that is July 19
- Cancel August 16 meeting?

About this Virtual Meeting

- Meeting info:
  - https://www.w3.org/2011/04/webrtc/wiki/May_17_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/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/mediacapture-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 - 08:50 AM WebRTC-Extensions & WebRTC-PC (Bernard)
  - Slides (08:10 - 08:30)
  - Discussion (08:30- 08:50)
- 08:50 - 09:10 CaptureController (Jan-Ivar)
  - Slides (08:50 - 09:00)
  - Discussion (09:00 - 9:10)
- 09:10 - 09:30 WebRTC-encoded-transform and Mediacapture-Extensions (Youenn)
  - Slides (09:10 - 09:20)
  - Discussion (09:20 - 9:30)
- 09:30 - 09:50 Dynamic Sources while Screensharing (“Share this X instead”)
  - Slides (09:30 - 09:40)
  - Discussion (09:40 - 9:50)
- 9:50 - 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.

WebRTC-Extensions & WebRTC-PC (Bernard)

Start Time: 08:10 AM

End Time: 08:50 AM

For Discussion Today

- **WebRTC-Extensions**
  - Issue 47/PR 106: RTP Header Extension Encryption
- **WebRTC-PC Simulcast Issues**
  - Issue 2735: webrtc-pc does not specify what level of support is required for RFC 7728 (RTP pause)
  - 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 47: RTP Header Extension Encryption

- "Completely Encrypting RTP Header Extensions and Contributing Sources" has completed IETF Last Call.
- API proposal presented at September 15, 2020 WEBRTC WG meeting:
  - Encrypt RTP headers on all m= sections (within a BUNDLE) or none of them.
  - Always attempt to negotiate it (e.g. “a=cryptex” on all m-lines)
  - Add `RTCConfiguration.rtpHeaderEncryptionPolicy`
    - "negotiate": If other endpoint doesn’t support it, send unencrypted.
    - "require": If the other endpoint doesn’t support it, fail to negotiate (throw an exception on setRemoteDescription).
  - Add `RTCRtpTransceiver.rtpHeaderEncryptionNegotiated` flag.

Issue 47: RTP Header Extension Encryption (cont'd)

- IETF cryptex spec marks extension as "BUNDLE: Transport"
  - If "a=cryptex" attribute is present, must be in all m lines of the bundle group.
  - Does not require that all m-lines are identical
    - If there are multiple bundle groups
    - When not bundling

Issue 47: RTP Header Extension Encryption (cont'd)

- What happens if a browser receives this Offer? (browser will never generate such an Offer)

```
a=group:BUNDLE A B
a=group:BUNDLE C D
m=audio
a=cryptex
a=mid:A
m=audio
a=cryptex
a=mid:B
m=video
a=mid:C
m=video
a=mid:D
```

- Answer
  - If `RTCConfiguration.rtpHeaderEncryptionPolicy = "require"`, browser rejects the offer.
  - If `RTCConfiguration.rtpHeaderEncryptionPolicy = "negotiate"`, browser will send cryptex for audio but not video, will be ready to receive on all m-lines.

PR 106: RTP Header Extension Encryption

§ 11. RTP Header Extension Encryption

§ 11.1 `RTCRtpHeaderEncryptionPolicy` Enum

RTP header extension encryption policy affects whether RTP header extension encryption is negotiated if the remote endpoint does not support [CRYPTEX]. If the remote endpoint supports [CRYPTEX], all media streams are sent utilizing [CRYPTEX].

WebIDL

```
enum RTCRtpHeaderEncryptionPolicy {
"negotiate",
"require"
};
```

Enumeration description (non-normative)	
`negotiate`	Negotiate RTP header extension encryption as defined in [CRYPTEX]. If encryption cannot be negotiated, RTP header extensions are sent in the clear.
`require`	Require RTP header extension encryption. In [WEBRTC] Section 4.4.1.5, add the following check after Step 4.4.4: If *remote* is `true`, the *connection*'s `RTCRtpHeaderEncryptionPolicy` is `require` and the description does not support [CRYPTEX], then reject *p* with a newly created `InvalidAccessError` and abort these steps.

PR 106: RTP Header Extension Encryption (cont'd)

§ 11.2 RTCRtpTransceiver extensions

rtpHeaderEncryptionNegotiated defines whether the transceiver is sending enrypted RTP header extensions as defined in [CRYPTEX].

WebIDL

```
partial interface RTCRtpTransceiver {
  readonly attribute boolean rtpHeaderEncryptionNegotiated;
};
```

§ Attributes

rtpHeaderEncryptionNegotiated **of type Boolean, readonly, nullable**

The rtpHeaderEncryptionNegotiated attribute is indicates whether RTP header extension encryption has been negotiated. On getting, the attribute *MUST* return the value of the [[RtpHeaderEncryptionNegotiated]] slot. In [WEBRTC] Section 5.4, add the following step to "create an RTCRtpTransceiver": Let *transceiver* have a **[[RtpHeaderEncryptionNegotiated]]** internal slot, initialized to false.

PR 106: RTP Header Extension Encryption (cont'd)

§ 11.3 RTCConfiguration extensions

rtpHeaderEncryptionPolicy defines the policy for negotiation of RTP header encryption using [CRYPTEX].

WebIDL

```
partial dictionary RTCConfiguration {
  RTCRtpHeaderEncryptionPolicy rtpHeaderEncryptionPolicy = "negotiate";
};
```

§ **Dictionary RTCConfiguration Members**

rtpHeaderEncryptionPolicy **of type RTCRtpHeaderEncryptionPolicy**

(FEATURE AT RISK) ISSUE 2

rtpHeaderEncryptionPolicy is marked as a feature at risk, since there is no clear commitment from implementers.

Issue 2735: webrtc-pc does not specify what level of support is required for RFC 7728 (RTP pause)

webrtc-pc has support for pausing simulcast encodings, which RFC 8853 says requires support for RFC 7728, at least in an answer:

"An answerer that receives an offer without RTP stream pause/resume capability MUST NOT mark any simulcast streams as initially paused in the answer."

However, RFC 7728 allows partial implementation; there are 8 different "profiles" (that are negotiated with a=rtcp-fb:ccm pause config=<number>):

https://datatracker.ietf.org/doc/html/rfc7728#section-9

webrtc-pc probably ought to specify the minimum level of support for RFC 7728.

- Problem: RFC 7728 is not implemented
  - RFC 8853 does not mandate (or even recommend) support for RFC 7728, which is not implemented in any browser.
  - RFC 7728 is IPR encumbered.
  - It is not appropriate to add a requirement to support a non-implemented, encumbered protocol, as part of recycling WebRTC-PC at Recommendation.

Issue 2735bis: webrtc-pc specifies using '~' in a=simulcast, but does not support RFC 7728 (RTP pause), which is a prerequisite

Problem: WebRTC-PC does ***not*** mention use of '~'

- setParameters() definition in Section 5.2 specifically indicates that it does not cause SDP renegotiation (precluding changing parameters negotiated in SDP)

> `setParameters` does not cause SDP renegotiation and can only be used to change what the media stack is sending or receiving within the envelope negotiated by Offer/Answer. The attributes in the `RTCRtpSendParameters` dictionary are designed to not enable this, so attributes like `cname` that cannot be changed are read-only. Other things, like bitrate, are controlled using limits such as `maxBitrate`, where the user agent needs to ensure it does not exceed the maximum bitrate specified by `maxBitrate`, while at the same time making sure it satisfies constraints on bitrate specified in other places such as the SDP.

The Video Layer Allocation (VLA) RTP header extension indicates when simulcast layers are stopped (bandwidth = 0), so there is no need to for ~ in SDP.

More 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:

```
1771 +                          <p>If a suitable transceiver was found (<var>transceiver</var>
1772 +                          is set) and <var>sendEncodings</var> is non-empty, set
1773 +                          <var>transceiver</var>.<a>[[\Sender]]</a>.<a>[[\sendEncodings]]</a>
1774 +                          to <var>sendEncodings</var>, and set
1775 +                          <var>transceiver</var>.<a>[[\Sender]]</a>.<a>[[\LastReturnedParameters]]</a>
1776 +                          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: 08:50 AM)

-

CaptureController (Jan-Ivar)

Start Time: 08:50 AM

End Time: 09:10 AM

#190 Conditional Focus

Recap: getDisplayMedia *"enables the acquisition of a user's display"*. It's a **tool**.

But all browsers ***focus*** a window or tab ***upon capture***, which *assumes* the use case is traditional screen-sharing in a video conference (VC). This is too limiting for apps:

- Capture Handle Identity and Actions exist to let apps keep the user in the VC tab
- Too soon for screen recording apps: user hasn't hit "Record" yet (maybe later?)

So where to put a focus control API? Why not the video track? Two reasons:

1. MediaStreamTrack is a clonable & constrainable *media consumption* abstraction. One source → many consumers. Whereas a method like track.cropTo is per-clone, a **track.focus()**, say, would affect ALL clones, a leaky abstraction because the user's focus is not an inherent property of a single video track.
2. Would both the video and audio track have this property? Why (not)? Confusing

We've stepped out of media consumption to remote control. Need a higher level object

#57 .sendCaptureAction() misplaced

Similarly, track.sendCaptureAction, the API to send supported actions to the captured page, seems misplaced on the MediaStreamTrack. The reasons are the same:

1. MediaStreamTrack is a clonable & constrainable *media consumption* abstraction. One source → many consumers. Whereas methods like track.cropTo are per-clone, sendCaptureAction would affect ALL clones, a leaky abstraction because progression of the captured page isn't a property of a single video track.
2. Would both the video and audio track have this property? Why (not)? Confusing

We've stepped out of media consumption to remote control.

A higher level "controller" object seems needed, that is owned by the caller, isn't clonable or necessarily shared with every media consumer.

#12 .getCaptureHandle() misplaced?

BTW the same can be said for getCaptureHandle. 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.

But Chrome has already shipped this, so this proposal focuses instead on conditional focus and actions at the moment, even though identity is mentioned in subsequent slides.

#12 CaptureController

The dedicated controller object: *

```
const controller = new CaptureController();
const stream = await navigator.mediaDevices.getDisplayMedia({controller});

const {origin, handle} = controller; // identity #12

await controller.focus(); // Default is no focus when controller is added! #190

const actions = controller.getSupportedActions(); // actions #57
if (actions.includes("nextslide")) {
  await controller.sendAction("nextslide");
}
```

* For illustrative purposes. The controller is associated 1←→1 upon gDM success, and may not be reassociated. Presence of the controller suppresses the focus that happens today. Without a controller, gDM would continue to focus for backwards compatibility. Apps may call focus() anytime ahead of gDM or right after it (upto 1 second after seems fine, but let's iterate on details). focus() resolves after captured window or tab has focus or asap on monitors, but no earlier than gDM success. It rejects if gDM rejects. getSupportedActions() may be empty. Exclusions may apply. Limit one gDM per controller. Non-transferable. No cash value.

Discussion (End Time: 09:10 AM)

-

WebRTC-encoded-transform & Mediacapture-extensions (Youenn)

Start Time: 09:10 AM

End Time: 09:30 AM

PR 132: Make generateKeyFrame take a single RID or none. Resolve the promise with the frame timestamp

- Current API is taking several RID parameters
  - Allow generation of several key frames with one call
  - Promise returns the time of reception of the first key frame

- Useful to easily identify the corresponding video frame
  - Promise resolution helps for the first key frame
    - It would be nice to return the key frame timestamp
  - Difficult to handle in case of multiple RIDs provided

- Proposal: pass a single optional RID to generateKeyFrame

PR 132: Make generateKeyFrame take a single RID or none. Resolve the promise with the frame timestamp

- Proposal: pass a single optional RID
  - Old version

    ```
    Promise<undefined> generateKeyFrame(optional sequence <DOMString> rids);
    ```

  - New version

    ```
    Promise<unsigned long long> generateKeyFrame(optional DOMString rid);
    ```

  - Resolve the promise with key frame timestamp

- Old API can be shimed
  - `return Promise.race(rids.map(rid => transformer.generateKeyFrame(rid)));`

PR 61: Add support for background blur and configuration change event

- See also explainer
- Deployment of background blur in web pages
  - Implemented using web technology: WebGL…)
- Some OSes/cameras have background blur built-in support
  - More efficient than web technology based solution
    - 2x+ power improvement measurements (see explainer)
  - Redundant to have OS bg blur + web site bg blur
- Proposal: add a 'backgroundBlur' capability/setting
  - Identify whether background blur is available/on/off, allow switching on/off background blur
  - Start with boolean values, evaluate double values

PR 61: Add support for background blur and configuration change event

- Some OSes may restrict/change camera/microphone settings
  - Outside of UA control
  - User may decide to turn on background blur

- Proposal: add a configurationchange event
  - Notify web application of camera settings/capabilities change

```
const stream = await navigator.mediaDevices.getUserMedia({video: true});
const track = stream.getVideoTracks()[0];
track.onconfigurationchange = () => {
    // enable/disable web application background blur
    toggleBackgroundBlur(track, track.getSettings().backgroundBlur);
};
```

PR 59: Add powerEfficientPixelFormat constraint

- Some cameras generate MJPEG video frames
  - OS will decompress them before feeding User Agent
  - Potential power impact if MJPEG decoding is not HW backed

- Proposal: add a boolean 'powerEfficientPixelFormat' constraint
  - Exposed as capabilities and settings
  - Usable by getUserMedia and applyConstraints

Discussion (End Time: 9:30 AM)

-

Dynamic Source for Screensharing (Elad)

Start Time: 09:30 AM

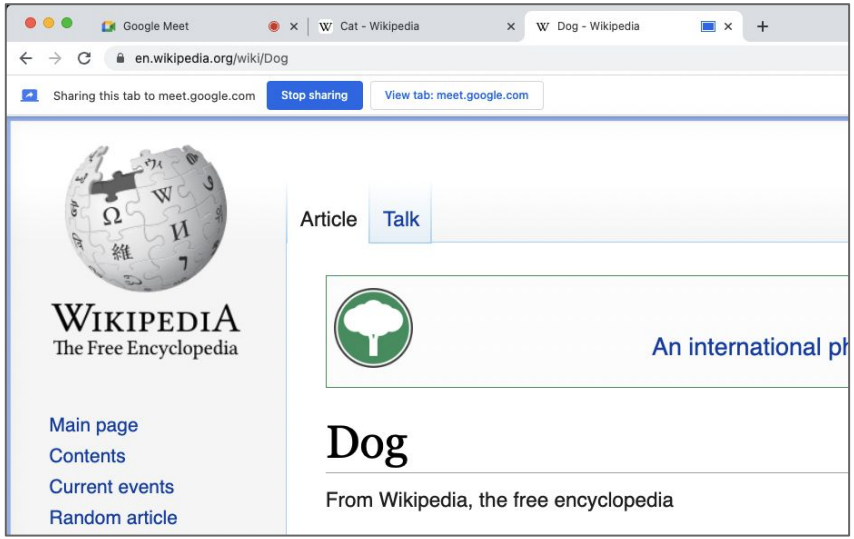
End Time: 09:50 AM

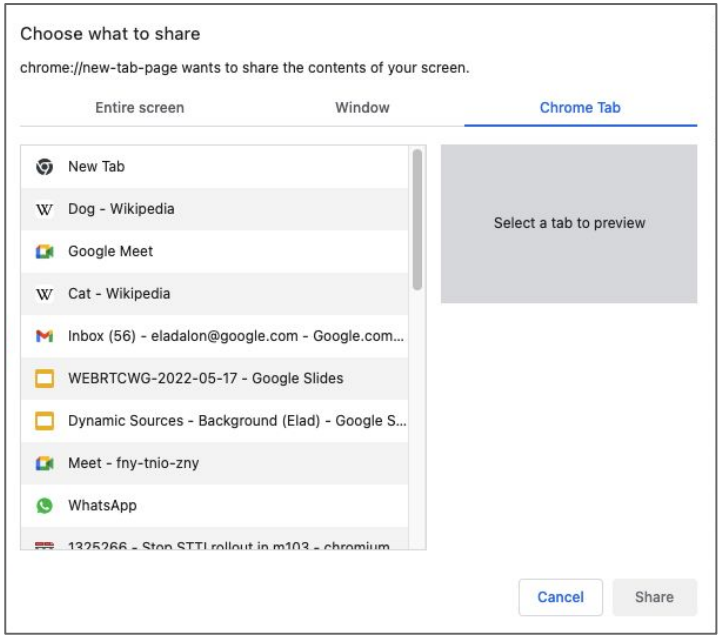
Introduction

- Users tend to share multiple tabs per session.
- Changing capture-source is ridiculously arduous.
- We can make it better.
  - There are challenges.
  - And solutions.

The Arduous Process of Changing Sources

- Find the video conferencing tab.
- Find the tab you want to share next.
- Maintain train of thought.

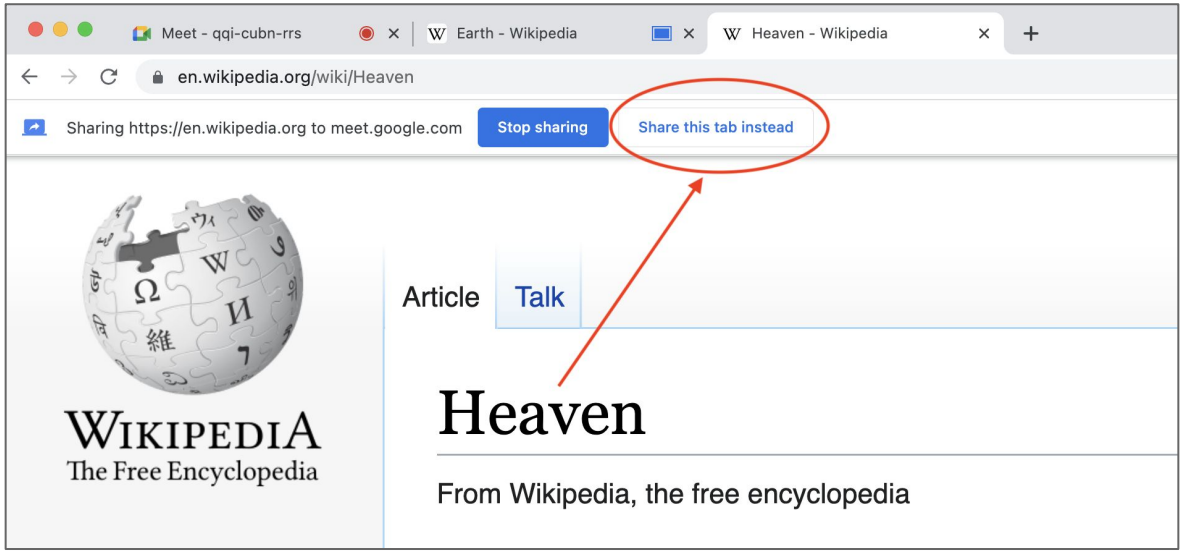




Utopia

Alternatives exist.

Deployment is challenging.



Deployment Challenges

Source-changing is new.

Existing applications are not adapted.

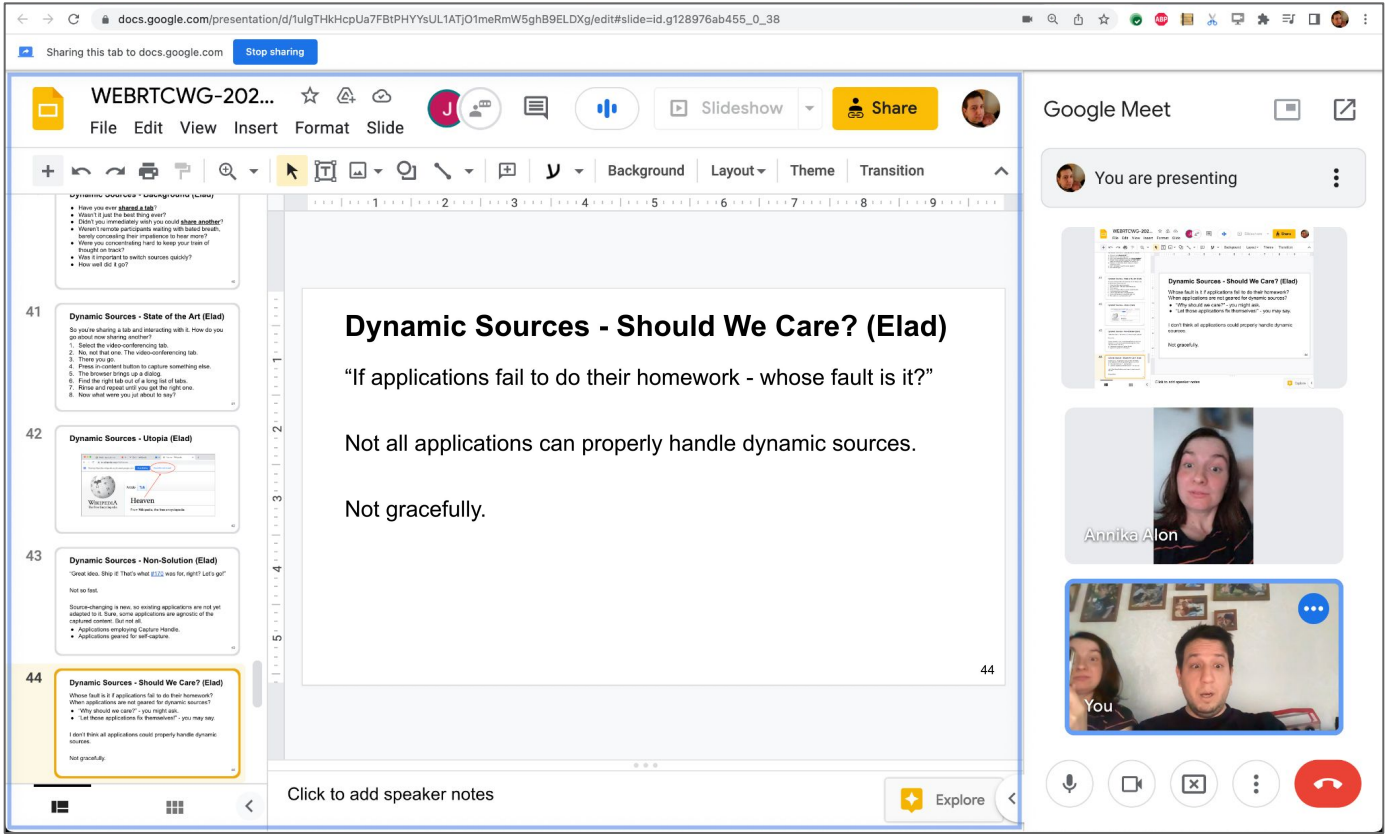
Some applications can’t ever fully adapt.

Granted, some applications are agnostic of the captured content.

But not all.

- Applications employing Capture Handle for remote-control.
- Applications centered around self-capture.

Case Study #1


docs.google.com/presentation/d/1ulgTHkHcpUa7FBtPHYYsUL1ATjO1meRmW5ghB9ELDXg/edit#slide=id.g128976ab455_0_38
Sharing this tab to docs.google.com
Stop sharing
WEBRTCWG-202...
File Edit View Insert Format Slide
Slideshow
Share
Background
Layout
Theme
Transition
41 Dynamic Sources - State of the Art (Elad)
42 Dynamic Sources - Utopia (Elad)
43 Dynamic Sources - Non-Solution (Elad)
44 Dynamic Sources - Should We Care? (Elad)
Dynamic Sources - Should We Care? (Elad)
"If applications fail to do their homework - whose fault is it?"
Not all applications can properly handle dynamic sources.
Not gracefully.
44
Click to add speaker notes
Explore
Google Meet
You are presenting
Annika Alon
You

Case Study #2

TBD: Screenshots portraying one application remote controlling another.

What if the user starts sharing another tab that cannot be remote controlled?

Show the user an error message?

Where would it even appear? The user won’t see it.

And would the user be happy if they did see it? Or…

Blame Attribution

- The browser displays a **big shiny button** for changing capture-source.
- The user presses the button.
- The difference between application and browser can be confusing.
- **“If I wasn’t supposed to press it, why did you put it there?”**

Solution

Let’s not guess what the application wants. Let it tell us.

```
dictionary DisplayMediaStreamConstraints {
  bool sourceChangeSupported;
};
```

Discussion (End Time: 9:50 AM)

Wrapup and Next Steps (End Time: 10:00 AM)

-

Thank you

Special thanks to:

WG Participants, Editors & Chairs