W3C WebRTC WG Meeting

September 25, 2018
1 PM - 2:30 PM Pacific Time

Chairs: Bernard Aboba
Harald Alvestrand

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 interim meeting of the W3C WebRTC WG!
- During this meeting, we hope to:
  - Introduce the “WebRTC Next Version Use Cases” document.
  - Make progress on open issues in screen sharing and webrtc-pc
- Will publish new editors drafts after the meeting

About this Virtual Meeting

Information on the meeting:

- Meeting info:
  - https://www.w3.org/2011/04/webrtc/wiki/Sept 25_2018
- Link to latest drafts:
  - https://w3c.github.io/mediacapture-main/
  - https://w3c.github.io/mediacapture-output/
  - https://w3c.github.io/mediacapture-screen-share/
  - https://w3c.github.io/webrtc-pc/
  - https://w3c.github.io/webrtc-stats/
  - https://www.w3.org/TR/mst-content-hint/
  - https://w3c.github.io/webrtc-nv-use-cases/
  - https://w3c.github.io/webrtc-dscp-exp/
- 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

A Word of Thanks

Best wishes to Stefan Hakansson, who after 7 years as a WEBRTC WG Chair has now stepped down.

Stefan’s contributions are much appreciated, and he will be greatly missed!

TPAC (Lyon, France)

We will meet Monday October 22 and Tuesday the 23rd of TPAC week.

Reviews are pouring in for….

WebRTC Next Version Use Cases:

https://w3c.github.io/webrtc-nv-use-cases/

“Like Moby Dick, without the whale.”

“It would have been a pleasure to burn, if it was worth printing.”

File your Issues today!

https://github.com/w3c/webrtc-nv-use-cases/issues

For Discussion Today

- mediacapture-screen-share Issues
  - Issue 28: Interaction with permissions API (Jan-Ivar)
  - Issue 61: Mention capture of system audio (henbos)
  - Issue 62: Offer high level source filtering (Jan-Ivar)
  - Issue 72: Describe what happens if the window is closed or the monitor is disconnected (henbos)
  - Issue 77: Should getDisplayMedia be functional in SecureContext only? (youennf)

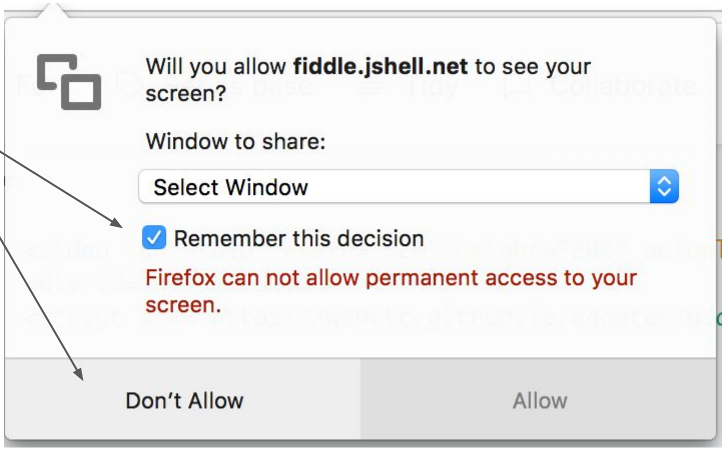
For Discussion Today (cont’d)

- **WebRTC-PC**
    - **[Issue 1858](): What happens when an Answerer stops a transceiver others are “bundled” on? (Bernard)**
    - **[Issue 1888](): RTCPriorityType is not documented at all (Harald)**
    - **[Issue 1896](): Order of RTCRtpSendParameters.encodings is not described (Harald)**
    - **[Issue 1930](): Rename sender.transport.transport to sender.transport.iceTransport? (Jan-Ivar)**
    - **[Issue 1982](): Missing normative steps for determining codecs (Jan-Ivar)**
    - **[Issue 1983](): getSynchronizationSources and getContributingSources should work for video too (henbos)**

Issue 28: Interaction with permissions API (Jan-Ivar)

- Persistent “granted” permissions are forbidden.
  Persistent “denied” permissions are allowed.

```
enum PermissionName {
  "camera",
  "microphone",
  "display",     // ← so we need this
  …
}
```



Will you allow **fiddle.jshell.net** to see your screen?

Window to share:

Select Window

✓ Remember this decision

Firefox can not allow permanent access to your screen.

...getDisplayMedia already assumes this (search for “display” with quotes), and

already says *“The User Agent MUST NOT create a permission storage entry with a value of "granted". ”*

- Covers access to both video and any related audio. No need to separate.
  I.e. if you block one you block both (is there even an audio-only use-case?)

- UA’s are responsible for explaining in prompts what’s being shared.

Issue 61: Mention capture of system audio (henbos)

- Support capturing audio. Use cases:
  - Sharing a video (including audio) in a presentation.
  - Listen to music together.
- Problem: System audio capabilities vary by OSes and implementations. Not clear if we can or want to include “window capture”, etc.

- Proposal: getDisplayMedia({audio:true}) asks for audio; up to implementation what audio sources are included.
  - Similar to how getUserMedia({audio:true}) does not limit input devices.

Issue 61: Mention capture of system audio (henbos) (cont.)

Remote participants hearing themselves in the screen capture makes the stream unusable.

- I previously proposed "excludeTabAudio" constraint as "all system audio minus application's audio". It would do the job, but may be technically infeasible.

- Proposal: {excludeApplicationAudio:true}
  - Implementation free to include any audio sources as long as the application's audio is not included.
  - "Captured tab's audio only" fits the bill.
  - ("All system audio minus application's audio" also fits the bill but is not a requirement.)
  - Downside: "no audio" also fits the bill, but this is akin to the user muting his or her microphone - perhaps it should be allowed? As long as we don't get echo, we're happy.

Issue 62: Offer high level source filtering (Jan-Ivar)

- **Request:** “Chrome and Firefox both offer the ability for the web app to restrict the sharing choice to desktop, window, tabs etc. Web apps like Google Hangouts use this functionality and would like to keep it”
- Spec expressly forbids it:
  - *”The user agent MUST let the end-user choose which display surface to share out of all available choices every time, and MUST NOT use constraints to limit that choice. Instead, constraints MUST be applied to the media chosen by the user, only after they have made their selection. This prevents an application from influencing the selection of sources”*
  - *“UAs are encouraged to warn users against sharing browser display devices and monitor display devices where browser windows are visible, or otherwise try to discourage their selection on the basis that these represent a significantly higher risk when shared.”*
- It’s why getDisplayMedia() didn’t allow constraints until a few months ago. Constraints were reinstated thanks to the above strong language.

Issue 62: Offer high level source filtering (cont’d)

- Reasons outlined in Security and Permissions:
  - *“Display capture presents a less obvious risk to the cross site request forgery protections offered by the browser sandbox. Display and capture of information that is also under the control of an application, even indirectly, can allow that application to access information that would otherwise by inaccessible to it directly.” [...]*

  - *“This issue is discussed in further detail in [RTCWEB-SECURITY-ARCH] and [RTCWEB-SECURITY]. Display capture that includes browser windows, particularly those that are under any form of control by the application, risks violation of these basic security protections.” [...] ”It is strongly advised that elevated permissions be required to access any display surface that might be used to circumvent cross-origin protections for content.”*
- Firefox plan is to remove “screen” vs “window” distinction in legacy API in transition.
- Proposal: close

Security concerns

Full-screen/browser sharing is scary!

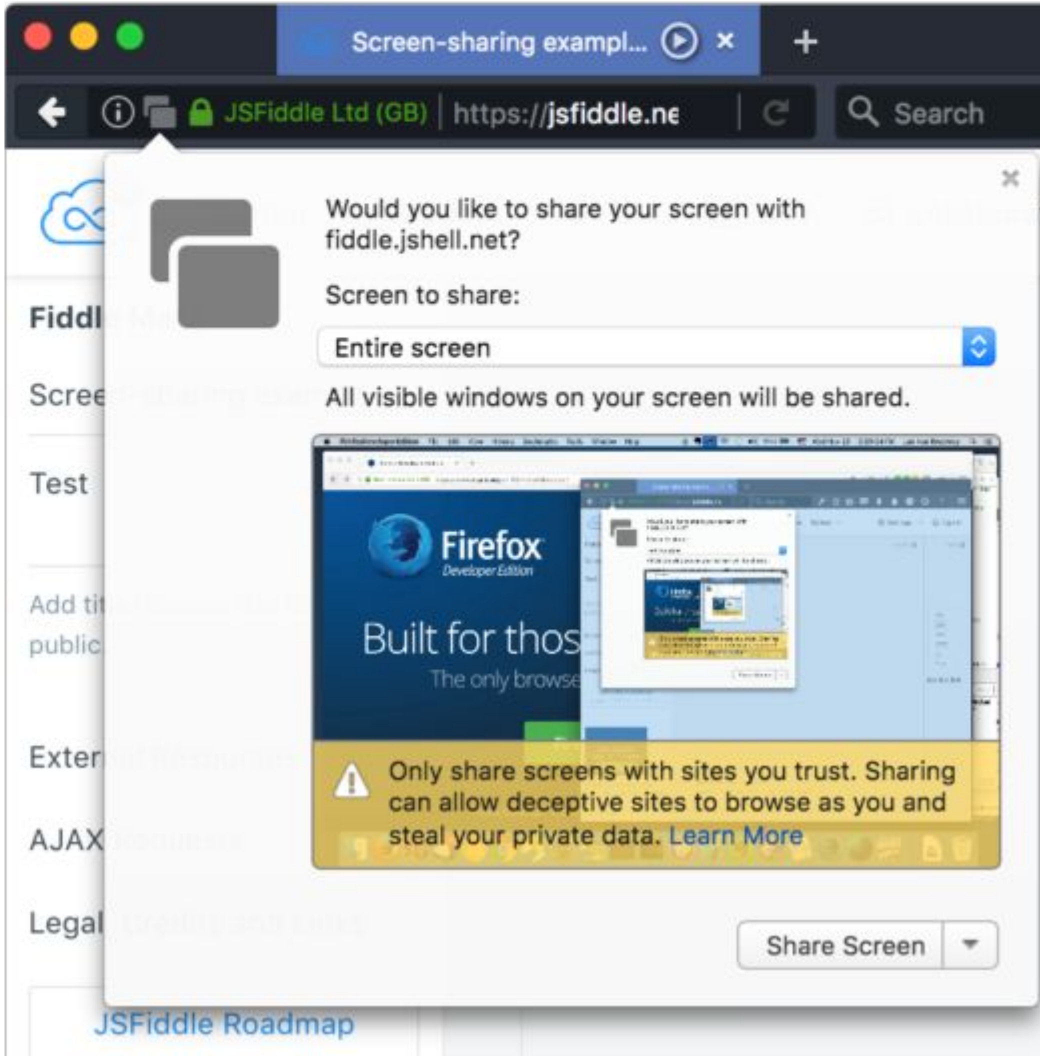
Not just passive threats.

If a web surface under site control is captured, that website has keys to the car, and can iframe-navigate as the logged-in user effectively.

Sidesteps cross-origin protections.

Firefox warns, but hard to explain ☞

Google “share screen trust” for more



Would you like to share your screen with fiddle.jshell.net?

Screen to share:

Entire screen

All visible windows on your screen will be shared.

⚠ Only share screens with sites you trust. Sharing can allow deceptive sites to browse as you and steal your private data. Learn More

Issue 62: Offer high level source filtering (Jan-Ivar)

That said, might it be OK to allow influencing *away* from “scary” sources?

```
await navigator.getDisplayMedia({video: {displaySurface: ”window”}});      // OK
await navigator.getDisplayMedia({video: {displaySurface: ”application”}}); // OK
await navigator.getDisplayMedia({video: {displaySurface: ”monitor”}}); //TypeError
await navigator.getDisplayMedia({video: {displaySurface: ”browser”}}); //TypeError
```

Any use-case for it? (sharing e.g. Google doc sadly is browser).

Is this a slippery slope?

Issue 72: Describe what happens if the window is closed or the monitor is disconnected (henbos)

The life-cycle of MediaStreamTracks lends itself getDisplayMedia(); mute/unmute is a temporary state, ended is permanent.

- Proposal: If window is closed or monitor disconnected, end the track. MediaStreamTrack.readyState = 'ended' and MediaStreamTrack.onended fires.

Minimized windows: “hidden” or special case of “covered”?

- Proposal A: If we want to hide it, mute/unmute the track. (MediaStreamTrack.muted, .onmute/onunmute)
- Proposal B: If we consider minimizing the same as covering the window, continue to capture it.

Issue 77: Should getDisplayMedia be functional in SecureContext only? (youennf)

- Option 1: reject promise for non-secure contexts
  - Consistent with getUserMedia
- Option 2: Expose getDisplayMedia for secure contexts only
  - Consistent with most other APIs
- Follow-up issue
  - https://github.com/w3c/mediacapture-main/issues/540

For Discussion Today

- WebRTC-PC
  - Issue 1858: What happens when an Answerer stops a transceiver others are “bundled” on? (Bernard)
  - Issue 1888: RTCPriorityType is not documented at all (Harald)
  - Issue 1896: Order of RTCRtpSendParameters.encodings is not described (Harald)
  - Issue 1930: Rename sender.transport.transport to sender.transport.iceTransport? (Jan-Ivar)
  - Issue 1982: Missing normative steps for determining codecs (Jan-Ivar)
  - Issue 1983: getSynchronizationSources and getContributingSources should work for video too (henbos)

Issue 1858: What happens when an answerer stops a transceiver that others are "bundled" on? (Bernard)

- If there's an established bundle group:
    - `a=group:BUNDLE audio video`
    - The first identification-tag in the bundle group is known as the "BUNDLE-tag", which identifies the tagged "m=" section.
- BUNDLE guidance:
    - Section 7.2.1: "It is RECOMMENDED that the suggested offerer tagged "m=" section is a bundled "m=" section that the offerer believes it is unlikely the answerer will reject, or move out of the BUNDLE group."
    - Section 7.3.3: "When an answerer wants to reject a bundled "m=" section in an answer, [if] In the corresponding offer, the "m=" section is the offerer tagged "m=" section… the answerer can not reject the "m=" section in the answer. The answerer can either reject the whole offer, reject each bundled "m=" section within the BUNDLE group, or keep the "m=" section within the BUNDLE group in the answer and later create an offer where the "m=" section is disabled within the BUNDLE group…"

Issue 1858: What happens when an answerer stops a transceiver that others are “bundled” on? (cont’d)

- So what happens if the answerer calls `audioTransceiver.stop()`?
    - The answerer can’t reject the “audio” m= section without rejecting “video” as well.
    - Both m= sections are rejected (both transceivers end up stopped).
- API limitations:
    - createOffer() text does not specify which transceiver is associated with an “offer-tagged” “m=section”.
        - Do all implementations use the initial transceiver?
    - Without parsing SDP, answerer cannot determine whether a transceiver is associated with an “offer-tagged” “m=section”.

Issue 1858: What happens when an answerer stops a transceiver that others are "bundled" on? (cont'd)

- Recommendation:
  - Add text stating that implementations should utilize the initial transceiver as the "bundle-tagged" "m=" section.
  - Add note explaining the potential side-effects of transceiver.stop().
  - Recommend that transceiver.stop() only be called on all bundled transceivers.

Issue 1888: RTCPriorityType is not documented for simulcast (Harald)

- RTCPriorityType is defined as priority between RTP streams coming from different sources. It is not defined between encodings in a simulcast.
- The congestion response part of Priority is intended to let all streams send some data. This is in conflict with normally-expected simulcast behavior, which drops layers on congestion.
- Quick fix #1: Claim that all encodings must have the same priority value (compatible, looks silly)
- Quick fix #2: Move priority to the Sender object (same effect, breaks compatibility)
- Slow fix #1: Define behavior of priority differences across encodings

Group preference: ?

Issue 1896: Order of RTCRtpSendParameters.encodings is not described (Harald)

- People have made different assumptions on order of encodings
- Some APIs imply differing treatment based on order
    - Section 5.1: If too many simulcast layers, layers truncated from the tail
    - Section 5.2: Encoding order changes prohibited in setParameters()
- Not clear that one suggested way is “the best one”
- Proposal:
    - State that order is consistent (starting from first)
    - State that simulcast layers are truncated from tail
    - State a rule (based on priority? See previous issue) on which order simulcast layers are dropped in when congestion happens (suggestion: last dropped first when priority equal)
    - Don’t require any particular ordering low > high or high > low

Issue 1930: Rename sender.transport.transport to sender.transport.iceTransport? (Jan-Ivar)

Two nested attributes of the same name is unintuitive / hard to read:

```
pc.getTransceivers()[0].sender.transport.transport; // whah?
```

Can we rename it?

```
pc.getTransceivers()[0].sender.transport.iceTransport; // ah!
```

Edge already implements this, thus would be affected.

But with WebRTC for ORTC already shimmed in adapter, is this fixable? Shim:

```
sender.transport.iceTransport || sender.transport.transport;
```

Issue 1982: Missing normative steps for determining codecs (Jan-Ivar)

sender.getParameters today:*"The codecs sequence is populated based on the codecs that have been negotiated for sending, and which the user agent is currently capable of sending"*
receiver.getParameters today: *"The codecs sequence is populated based on the codecs that the receiver is currently prepared to receive"*

But these are synchronous methods. Proposal:

sender.getParameters: "codecs *is set to the value of* ***[[SendCodecs]]***"
receiver.getParameters "codecs *is set to the value of* ***[[ReceiveCodecs]]***"

...and have SRD(Answer) and SLD(Answer):

"Set ***[[SendCodecs]]*** *to the codecs that have been negotiated for sending, and which the user agent is currently capable of sending" and*
"Set ***[[ReceiveCodecs]]*** *to the codecs that have been negotiated for receiving, and which the user agent is currently prepared to receive" ?*

```
E.g.:
    console.log(sender.getParameters().codecs.length); // 0
     await pc.setRemoteDescription(msg.offer);
     console.log(sender.getParameters().codecs.length); // 0
     await pc.setLocalDescription(await pc.createAnswer());
     console.log(sender.getParameters().codecs.length); // 3
```

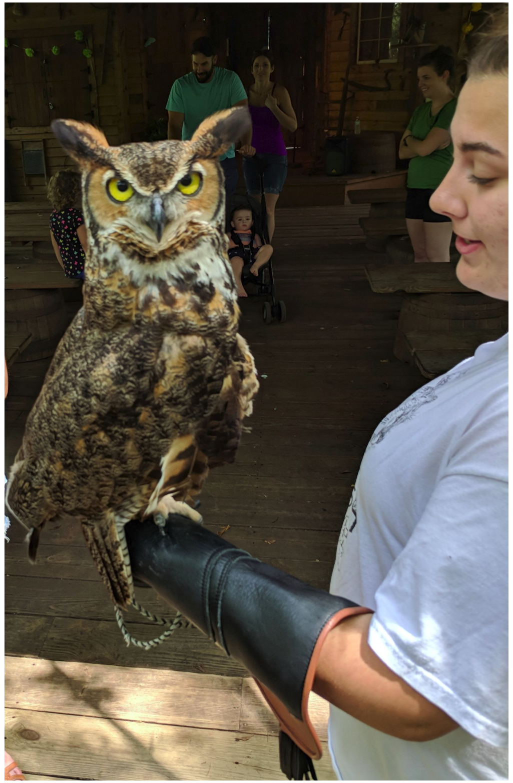
Issue 1983: getSynchronizationSources and getContributingSources should work for video too (henbos)

getSynchronizationSources() and getContributingSources() return the timestamps of the most recent RTP packets. Currently audio-only as a means to surface audioLevel.

- Can be used to check activity of a receiver. (Is RTP stream active?) This is useful for video too.
  - MediaStreamTrack “muted” state does not help. getStats() and packet counters is too heavy and clumsy, not suitable for real-time.
- CSRCs included by audio mixers, but also potentially usable by video MCUs.

- Proposal: Define getSynchronizationSources() and getContributingSources() for both audio and video.
  - Simply make audioLevel and voiceActivityFlag audio-only members (or members of child dictionaries).

For extra credit



Name that bird!

Thank you

Special thanks to:

W3C/MIT for WebEx

WG Participants, Editors & Chairs

The bird