W3C WebRTC WG Meeting

March 15, 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 March 2022 interim meeting of the W3C WebRTC WG, at which we will cover:
  - Avoiding the “Hall of Mirrors”
  - getViewportMedia
  - WebRTC-SVC
  - WebRTC-Extensions
  - MediaCapture-Extensions
- Future meetings:
  - April 19
  - May 17

About this Virtual Meeting

- Meeting info:
  - https://www.w3.org/2011/04/webrtc/wiki/March_15_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-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/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 being recorded. The recording will be public.

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.
  - It is possible to merge PRs that may lack consensus, if a note is attached indicating controversy.

Poll about TPAC 2022

- Are you considering attending in person TPAC 2022 (week of Sep 12 2022 in Vancouver)?
  - Yes
  - No
  - Don’t know

Issues for Discussion Today

- 08:10 - 08:40 AM (WebRTC-SVC and WebRTC-Extensions, Bernard)
  - Slides (08:10 - 08:25)
  - Discussion (08:25 - 08:40)
- 08:40 - 09:10 AM (Avoiding the “Hall of Mirrors”, Elad)
  - Slides (08:40 - 08:55)
  - Discussion (08:55 - 09:10)
- 09:10 - 09:20 AM (Display Surface Hints, getViewportMedia, Elad + Jan-Ivar)
- 09:20 - 09:50 AM: (MediaCapture-Extensions, Riju)
  - Slides (9:20 - 9:35 AM)
  - Discussion (9:35 - 9:50)
- 09:50 AM - 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-SVC & WebRTC-Extensions (Bernard)

Start Time: 8:10 AM

End Time: 8:40 AM

Issues/PRs for Discussion

WebRTC-SVC

- Issue 68/PR 69: Clarify behavior of getParameters()

WebRTC-Extensions

- Issue 47: RTP Header Extension Encryption
- Issue 98: Disabling hardware acceleration
- Issue 99: Should RTCRtpHeaderExtensionCapabilities offer an “enabled” member?

Issue 68: Clarify behavior of getParameters()

- Section 4.2.3 is unclear about re-negotiation:

Before negotiation has completed, getParameters() returns the scalabilityMode value for each encoding in encodings, assuming it was successfully set by addTransceiver() or setParameters(). If no scalabilityMode value was provided for an encoding in encodings, or if a value was not successfully set, then getParameters() will not return a scalabilityMode value for that encoding.

After negotiation has completed, getParameters() returns the currently configured scalabilityMode value for each encoding in encodings. This may be different from the values requested in addTransceiver() or setParameters(). If the configuration is not satisfactory, setParameters() can be used to change it.

If an encoding in encodings had no scalabilityMode value provided to addTransceiver() or setParameters(), getParameters() returns the default scalabilityMode of the most preferred codec. The most preferred codec and the default scalabilityMode for each ecodec are both implementation dependent. The default scalabilityMode *SHOULD* be one of the temporal scalability modes (e.g. "L1T1","L1T2","L1T3", etc.).

PR 69: Clarify behavior of getParameters()

- Proposed new language:

Before the initial negotiation has completed, getParameters() returns the scalabilityMode value for each encoding in encodings, as last set by addTransceiver() or setParameters(). If no scalabilityMode value was provided for an encoding in encodings, or if a value was not successfully set, then getParameters() will not return a scalabilityMode value for that encoding.

After the initial negotiation has completed, getParameters() returns the currently configured scalabilityMode value for each encoding in encodings. This may be different from the values requested in addTransceiver() or setParameters(). If the configuration is not satisfactory, setParameters() can be used to change it.

If addTransceiver() or setParameters() did not provide a scalabilityMode value for an encoding in encodings, then after the initial negotiation has completed, getParameters() returns the default scalabilityMode of the most preferred codec for that encoding. The most preferred codec and the default scalabilityMode for each ecodec are both implementation dependent. The default scalabilityMode *SHOULD* be one of the temporal scalability modes (e.g. "L1T1","L1T2","L1T3", etc.).

Issue 47: RTP Header Extension Encryption

- "Completely Encrypting RTP Header Extensions and Contributing Sources" has completed WGLC in IETF AVTCORE WG.
- 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.
  - 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.encryptionNegotiated` flag.
- Problem: IETF cryptex spec marks extension as "BUNDLE: Transport"
  - If "a=cryptex" attribute is present, must be in all m lines of the bundle group.
    - But all m-lines need not be identical (e.g. when not bundling).

Issue 47: RTP Header Extension Encryption (2)

- Harald: Creates interop problem with non-browsers. Without “IDENTICAL” we can’t reject this Offer (cryptex on audio but not video):

```
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
```

Issue 98: Disabling hardware acceleration

- Fippo has provided a long list of hardware-acceleration implementation bugs. A sample:
  - Hardware H264 Encoder is being used instead of OpenH264 on MacOS resulting in poor frame rate / pixelated video: https://bugs.chromium.org/p/chromium/issues/detail?id=1142273&
  - Resolution cannot ramp-up to 720P on Windows Chrome 91 with HW acceleration enabled: https://bugs.chromium.org/p/webrtc/issues/detail?id=12942
  - VP8 encoder issue when hardware acceleration enabled on mobile device: https://bugs.chromium.org/p/chromium/issues/detail?id=1237677
  - Windows H264 encoder may freeze: https://bugs.chromium.org/p/chromium/issues/detail?id=1252710
- Can we provide a way to disable hardware-acceleration?
  - WebCodecs has VideoEncoderConfig.hardwareAcceleration:

```
enum HardwareAcceleration {
  "no-preference",
  "prefer-hardware",
  "prefer-software",
};
```

Issue 98: Potential Approach

- RTCRtpTransceiver.setCodecPreferences()
  - Extend RTCRtpCodecCapability dictionary:

```
partial dictionary RTCRtpCodecCapability {
   HardwareAcceleration hardwareAcceleration = "no-preference";

};
```

  - Influences codec/profile combinations in createOffer/createAnswer.
    - A hardware-only codec/profile combination would not surface in createOffer/createAnswer if hardwareAcceleration was set to “prefer-software”.
- RTCRtpCodecCapability discovery in Media Capabilities API
    - Issue 185: Retrieving RTCRtpCodecCapability from MediaCapabilities when queried for webrtc
      - Should a hardwareAcceleration member be returned to indicate software-only or hardware-only codecs?
      - Or is smooth/power efficient/supported enough?

Issue 99: Should RTCRtpHeaderExtensionCapabilities offer an “enabled” member? (Harald)

- Scenario: Implementation supports snazzy-extension
  - By default, it is not set in offers
  - It is listed in Capabilities
  - But doesn’t turn up on offers
  - You can’t see why
- Is this a problem?
  - If yes - make info visibie
  - If no (we can always inspect the offer) - no change

Discussion (End Time: 8:40 AM)

-

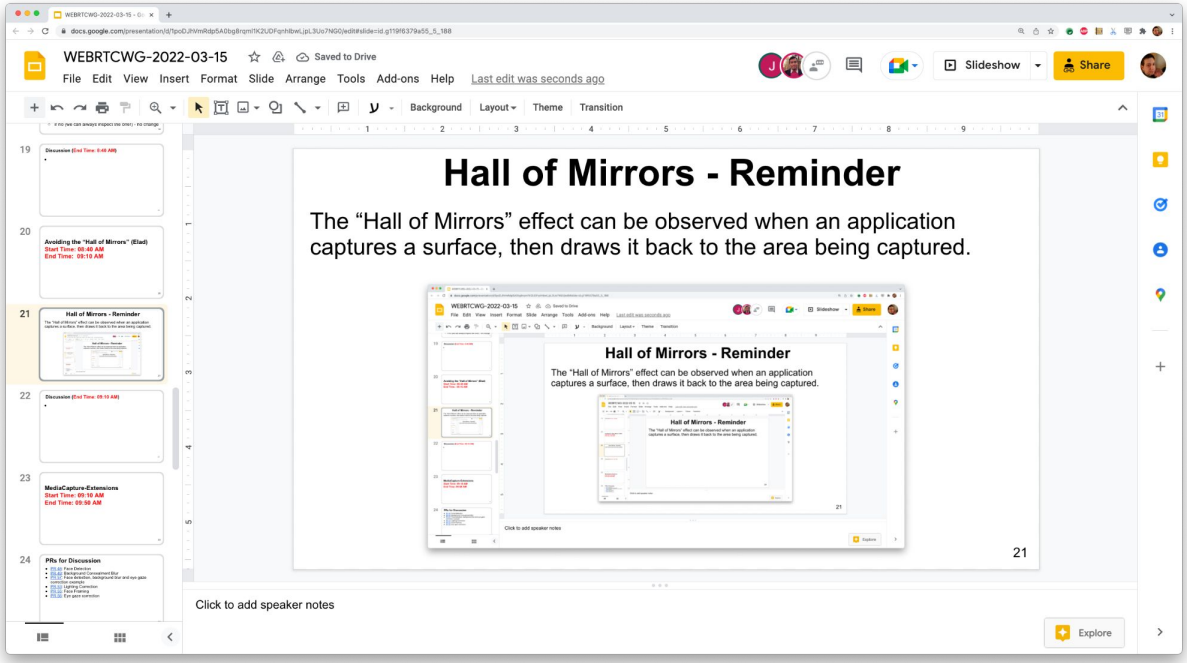
Avoiding the “Hall of Mirrors” (Elad)

Start Time: 08:40 AM

End Time: 09:10 AM

Hall of Mirrors - Reminder (Elad)

The “Hall of Mirrors” effect can be observed when an application captures a surface, then draws it back to the area being captured.



Hall of Mirrors - Problem (Elad)

Users can accidentally trigger one of three variants of the HoM effect:

1. Capturing the current-tab.
2. Capturing the current-window.
3. Capturing the current-screen.

Many video-conferencing applications display a preview of the captured content back to the user, triggering the HoM.

Adverse effects include:

1. Confuses local user.
2. Confuses remote users.
3. Could potentially produce mic-howl.

Hall of Mirrors - Rejected “Solution” (Elad)

Rejected: “Maybe the user agent should just exclude the current tab from the list of available tabs?”

Note that this (rejected) solution exclusively focuses on tab-capture.

This would break legitimate applications which intentionally record the current tab, for example recording gameplay or video demonstrations. Such applications would normally not show the local user a preview, which means they don’t have a problem with HoM.

Hall of Mirrors - Suggested Solution (Elad)

Recall that getDisplayMedia() receives the following dictionary.

We could extend it:

```
dictionary DisplayMediaStreamConstraints {
  (boolean or MediaTrackConstraints) video = true;
  (boolean or MediaTrackConstraints) audio = false;
  boolean includeCurrentTab = true;
};
```

The default behavior remains unchanged (in all browsers currently supporting tab-capture). But applications can trigger the new behavior, which avoids HoM by denying the user the possibility of selecting the current tab.

Hall of Mirrors - Security Discussion (Elad)

“That influences user choice! It could be abused for social engineering!”

We have discussed (often) the dangers of allowing an application to push the user towards either:

1. A display surface under the application’s control (e.g. current tab).
2. A display surface that is inherently dangerous (e.g. a screen).

The suggested solution does neither. There is no degradation in security.

Hall of Mirrors - Default Value (Elad)

Three options for a default value:

1. Avoiding breaking current applications. (includeCurrentTab: true)
2. Get us where we want to end up. (Open for discussion where that is…)
3. Don’t specify a default value.

I suggest we go with #3. This is a **hint**. It may be provided or omitted, and the user agent MAY regard or ignore it.

`If a value for includeCurrentTab is specified, it is a hint. The user agent MAY regard this hint when deciding whether to include the current tab in the list of surfaces it offers to the user.`

Hall of Mirrors - Potential Scope Expansion (Elad)

It is possible to go beyond tab-capture.

- Three distinct knobs:
  - includeCurrentTab
  - includeCurrentWindow
  - includeCurrentMonitor
- Cumulative knob:
  - includeCurrentTab
  - includeCurrentTabAndWindow
  - includeCurrentTabAndWindowAndMonitor

The security argument made in an earlier slide still holds, albeit less patently.

Discussion (End Time: 09:10 AM)

-

Display Surface Hints (Elad)
getViewportMedia (Jan-Ivar)
Start Time: 09:10 AM
End Time: 09:20 AM

Display Surface Hinting - Let’s Resolve (Elad)

We have been discussing displaySurface-hints for a long. The latest manigestation of this discussion is issue #184, which has been under discussion for 9 months by now.

This is a highly-requested control knob. We should be able to accommodate Web-developers in a timely manner with such small changes. We should converge on the least controversial proposal.

- Keep using established mechanisms - constraints.
- Leave the ultimate decision to the user agent. Keep it a hint.

User agents MAY change the order or prominence of offered choices in response to an application's preference, as indicated by the {{displaySurface}} constraint.

Let’s ship it.

getViewportMedia() update: ready for Call for Adoption

Document up at https://w3c.github.io/mediacapture-viewport/ (UD)

WebIDL

```
partial interface MediaDevices {
  Promise<MediaStream> getViewportMedia(optional DisplayMediaStreamConstraints constraints = {});
};
```

Recap: (resolutions from April & September)

- Captures top-level browsing context's viewport (current tab) even from an iframe
- Gated by
  - window.crossOriginIsolated
  - "Document-Policy: **viewport-capture**" (opt-in) + "Require-Document-Policy: **viewport-capture**" (#4)
  - User Permission "**viewport-capture**"
  - Permissions policy "**viewport-capture**" (in iframes)
  - transient activation
- Same privacy indicator requirements and constraints (video+audio) as getDisplayMedia

Discussion (End Time: 09:20 AM)

-

MediaCapture-Extensions

Start Time: 09:20 AM

End Time: 09:50 AM

PRs for Discussion

- PR 48: Face Detection
- PR 49: Background Concealment Blur
- PR 57: Face detection, background blur and eye gaze correction example
- PR 53: Lighting Correction
- PR 55: Face Framing
- PR 56: Eye gaze correction

PR 48: Face detection

```
partial interface VideoFrame {
  readonly attribute FrozenArray<DetectedFace>?
detectedFaces;

 };

dictionary DetectedFace {
  required long                     id;
  required float                    probability;
  FrozenArray<Point2D>              contour;
  FrozenArray<Point2D>              mesh;
  FrozenArray<DetectedFaceLandmark> landmarks;
};


 dictionary DetectedFaceLandmark {
   required FrozenArray<Point2D> contour;
   FaceLandmark                  type;

  };
 enum FaceLandmark {
   "eye", "eyeLeft", "eyeRight", "mouth","nose"};
```

```
partial dictionary MediaTrackSupportedConstraints
{
  boolean faceDetectionMode = true;
  boolean faceDetectionLandmarks = true;
  boolean faceDetectionMaxNumFaces = true;
  boolean faceDetectionNumContourPoints = true;
  boolean faceDetectionNumLandmarkPoints = true;

 };


partial dictionary MediaTrackCapabilities {
  sequence<DOMString> faceDetectionMode;
  sequence<boolean>   faceDetectionLandmarks;
  ULongRange          faceDetectionMaxNumFaces;
  ULongRange          faceDetectionNumContourPoints;
  ULongRange          faceDetectionNumLandmarkPoints;

 };
```

```
enum FaceDetectionMode {
  "none", "presence", "contour", "mesh"};
```

Face detection example

```
// Check if face detection is supported by the browser
const supports =
navigator.mediaDevices.getSupportedConstraints();
if (supports.faceDetectionMode &&
    supports.faceDetectionNumContourPoints) {
  // Browser supports face contour detection.
} else {
  throw('Face contour detection is not supported');
}

// Open camera with face detection enabled
const stream = await
navigator.mediaDevices.getUserMedia({
  video: {
    faceDetectionMode: 'contour',
    faceDetectionNumContourPoints: {exact: 4}
  }
});
const [videoTrack] = stream.getVideoTracks();
```

```
// Use a video worker and show to user.
const videoElement = document.querySelector('video');
const videoWorker = new Worker('video-worker.js');
videoWorker.postMessage({track: videoTrack}, [videoTrack]);
const {data} = await new Promise(r => videoWorker.onmessage);
videoElement.srcObject = new MediaStream([data.videoTrack]);

// video-worker.js:
self.onmessage = async ({data: {track}}) => {
  const generator = new VideoTrackGenerator();
  parent.postMessage({videoTrack: generator.track},
[generator.track]);
  const {readable} = new MediaStreamTrackProcessor({track});
  const transformer = new TransformStream({
    async transform(frame, controller) {
      for (const face of frame.detectedFaces) {
        console.log(
          `Face @ (${face.contour[0].x}, ${face.contour[0].y}), `
                `(${face.contour[1].x}, ${face.contour[1].y}), `
                `(${face.contour[2].x}, ${face.contour[2].y}), `
                `(${face.contour[3].x}, ${face.contour[3].y})`);
        controller.enqueue(frame);
      }
    }
  });
  await readable.pipeThrough(transformer).pipeTo(generator.writab
};
```

PR 49: Background Concealment Blur

```
partial dictionary
MediaTrackSupportedConstraints {
  boolean backgroundBlur = true;
};

partial dictionary
MediaTrackCapabilities {
  MediaSettingsRange backgroundBlur;
};

partial dictionary MediaTrackConstraintSet {
  MediaSettingsRange backgroundBlur;
};

partial dictionary MediaTrackSettings {
  double backgroundBlur;
};
```

value of 0.0 indicates no background blur and
increasing values indicate increasing background blur

```
const stream = await
 navigator.mediaDevices.getUserMedia({video: true});
const [videoTrack] = stream.getVideoTracks();

// Try to conceal background.
const videoCapabilities = videoTrack.getCapabilities();
if (videoCapabilities.backgroundBlur) {
  await videoTrack.applyConstraints({
    advanced: [{backgroundBlur:
videoCapabilities.backgroundBlur.max}]
  });
} else {
  // Background concealment is not supported by the platform
or by the camera.
  // Consider falling back to some other method.
}

// Show to user.
const videoElement = document.querySelector("video");
videoElement.srcObject = stream;
```

PR 57: Face detection, BG blur, etc example

```
const generator = new VideoTrackGenerator();
  parent.postMessage({videoTrack: generator.track}, [generator.track]);
  const {readable} = new MediaStreamTrackProcessor({track});
  const transformer = new TransformStream({
    async transform(frame, controller) {
      // Detect faces or retrieve detected faces.
      const detectedFaces =
        customFaceDetection
          ? await detectFaces(frame)
          : frame.detectedFaces;
      // Blur the background if needed.
      if (customBackgroundBlur) {
        const newFrame = await blurBackground(frame, detectedFaces);
        frame.close();
        frame = newFrame;
      }
      // Correct the eye gaze if needed.
      if (customEyeGazeCorrection && (detectedFaces || []).length > 0) {
        const newFrame = await correctEyeGaze(frame, detectedFaces);
        frame.close();
        frame = newFrame;
      }
      controller.enqueue(frame);
    }
  });
  await readable.pipeThrough(transformer).pipeTo(generator.writable);
};
```

LINK

PR 53: Lighting Correction

```
partial dictionary
MediaTrackSupportedConstraints {
  boolean lightingCorrection = true;

 };

partial dictionary MediaTrackCapabilities {
  sequence<boolean> lightingCorrection;

 };

partial dictionary MediaTrackConstraintSet {
  ConstrainBoolean lightingCorrection;

 };

partial dictionary MediaTrackSettings {
  boolean lightingCorrection;

 };
```

Lighting correction is a boolean setting controlling whether face and background lighting balance is to be corrected.

PR 55: Face Framing

```
partial dictionary
MediaTrackSupportedConstraints {
  boolean faceFraming = true;
};

partial dictionary MediaTrackCapabilities {
  sequence<boolean> faceFraming;
};

partial dictionary MediaTrackConstraintSet {
  ConstrainBoolean faceFraming;
};

partial dictionary MediaTrackSettings {
 boolean faceFraming;
};
```

Face framing is a boolean setting controlling whether framing is to be improved by cropping to faces.

PR 56: Eye gaze correction

```
partial dictionary
MediaTrackSupportedConstraints {
  boolean eyeGazeCorrection = true;
};

partial dictionary
MediaTrackCapabilities {
  sequence<boolean> eyeGazeCorrection;
};

partial dictionary
MediaTrackConstraintSet {
  ConstrainBoolean eyeGazeCorrection;
};

partial dictionary
MediaTrackSettings {
  boolean eyeGazeCorrection;
};
```

Eye gaze correction is a boolean setting controlling whether the eye gaze is to be corrected.

Discussion (End Time: 09:50 AM)

-

Thank you

Special thanks to:

WG Participants, Editors & Chairs