W3C WebRTC WG Meeting

June 23, 2022
7 AM - 9 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 June 2022 interim meeting of the W3C WebRTC WG, at which we will cover:
    - WebRTC WG Rechartering
    - Region Capture
    - Region Capture Extensions
    - Face Detection
- Future meetings:
    - July 19
    - September 12, 13, 15 (TPAC 2022)

About this Virtual Meeting

- Meeting info:
    - https://www.w3.org/2011/04/webrtc/wiki/June_23_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

- 07:10 - 07:20 WebRTC WG Re-Charter (Dom)
- 07:20 - 08:10 AM Region Capture Issues
- 08:10 - 08:35 AM Region Capture Extensions
- 08:35 AM - 08:55 AM Face Detection
- 08:55 AM - 09:00 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.

W3C WebRTC WG Re-Charter (Dom)

- Current charter expires end of September
- Draft of new charter
  - Mostly continuation of current work
  - Advisory Committee review in August
- Issue #70
  - Any spec to abandon or transfer to another group?

Region Capture Issues

Start Time: 07:20 AM

End Time: 08:10 AM

CropTarget.fromElement Case for Sync, Jan-Ivar

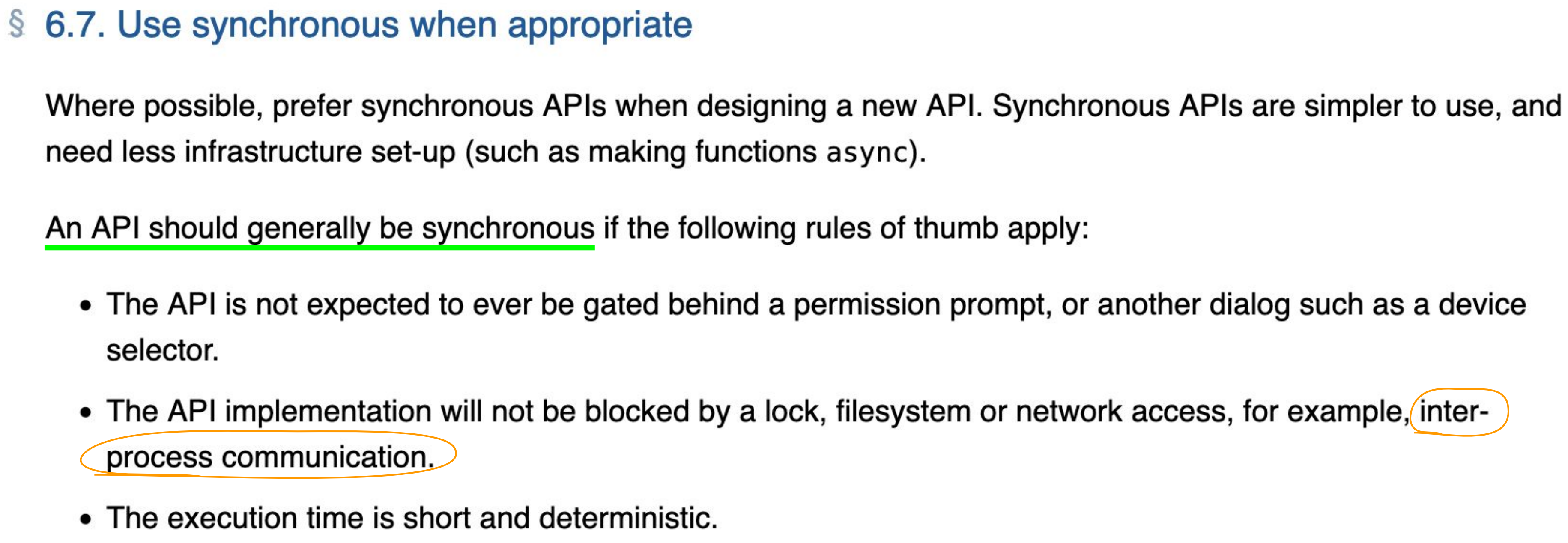
Start Time: 07:20 AM

End Time: 07:35 AM

Issues for Discussion Today

- Issue 17: What makes CropTarget special to require an asynchronous creation?
  - Sync Proposal: Jan-Ivar (10+5 min)
  - Async Proposal: Elad (10+5 min)
  - Shared discussion (10min)
- Issue 18: Is CropTarget name too generic? (Youenn 10min)

Issue 17: What makes CropTarget special to require an asynchronous creation? SYNC PROPOSAL (Jan-Ivar 1/6)



§ 6.7. Use synchronous when appropriate

Where possible, prefer synchronous APIs when designing a new API. Synchronous APIs are simpler to use, and need less infrastructure set-up (such as making functions `async`).

An API should generally be synchronous if the following rules of thumb apply:

- The API is not expected to ever be gated behind a permission prompt, or another dialog such as a device selector.
- The API implementation will not be blocked by a lock, filesystem or network access, for example, inter-process communication.
- The execution time is short and deterministic.

📙 https://www.w3.org/TR/design-principles/#synchronous

Issue 17: Why async CropTarget creation? (Jan-Ivar 2/6)

Non-consensus API is async: `const target = await CropTarget.fromElement(element);`

Proposed API is sync: `const target = new CropTarget(element);`

Its purpose: associate a serializable identifier with an element (aka minting)

> "Calling fromElement with an Element of a supported type associates that Element with a CropTarget.
> CropTarget is an intentionally empty, opaque identifier. Its purpose is to be handed to cropTo as input."

As currently specified, this ***cannot fail*** for non-synchronous reasons. But see #48.

CropTarget only exists because Element cannot be postMessaged:

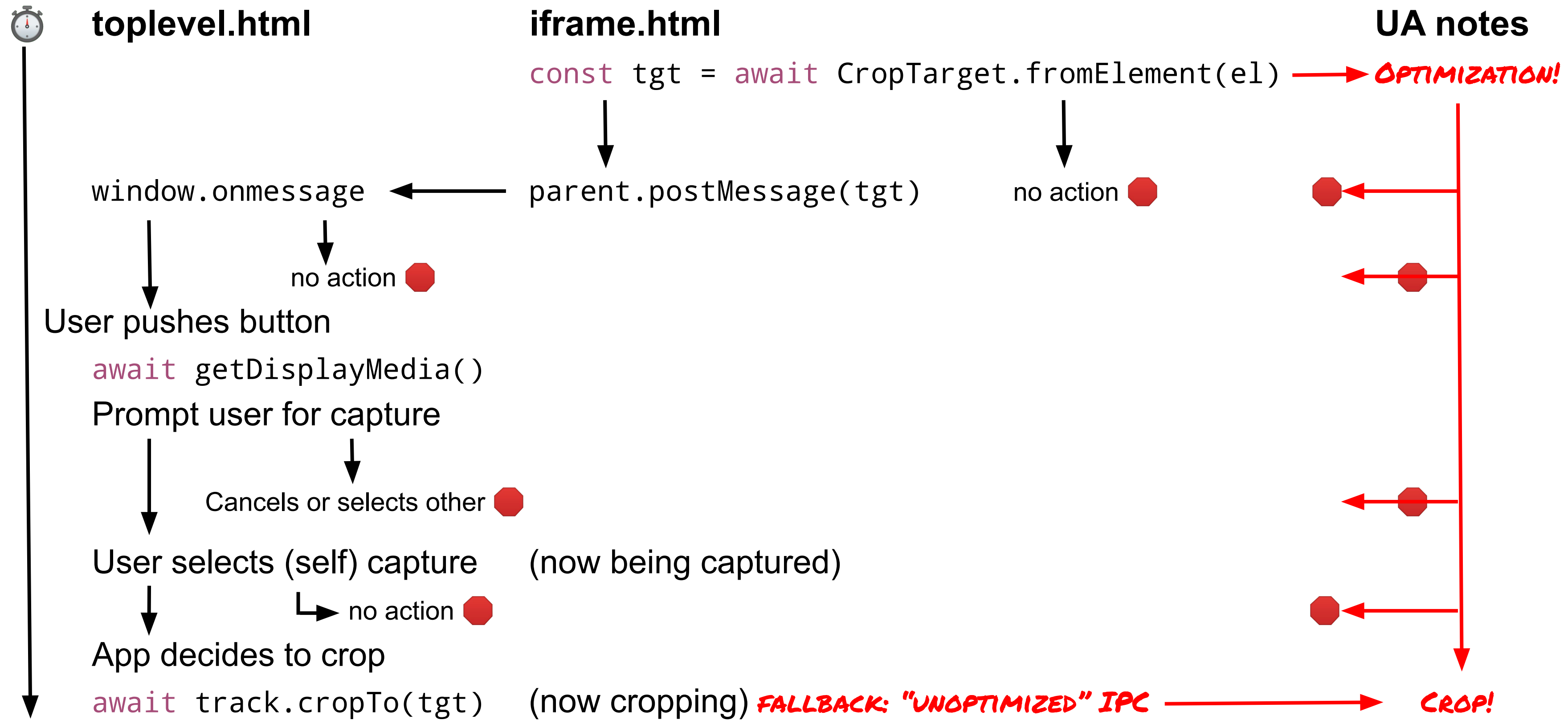
```
// iframe.html
parent.postMessage(new CropTarget(element), "*");

// toplevel.html
window.onmessage = async ({data: cropTarget}) => await track.cropTo(cropTarget);
```

Testable requirement: cropTo MUST accept it, which doesn't require BOTH to be async

Issue 17: Why async CropTarget creation? (Jan-Ivar 3/6)

Multiple actions need to happen before we're cropping anything. UAs can jump the gun but as an *optimization*



toplevel.html | **iframe.html** | **UA notes**

```
const tgt = await CropTarget.fromElement(el)
```

OPTIMIZATION!

```
window.onmessage   ←   parent.postMessage(tgt)
```

no action 🛑

no action 🛑

User pushes button

```
await getDisplayMedia()
```

Prompt user for capture

Cancels or selects other 🛑

User selects (self) capture — (now being captured)

no action 🛑

App decides to crop

```
await track.cropTo(tgt)
```

(now cropping) FALLBACK: "UNOPTIMIZED" IPC → CROP!

Issue 17: Why async CropTarget creation? (Jan-Ivar 4/6)

#48 is a new request on the spec to expose **resource exhaustion errors** in `fromElement`

But allowing random JS in would-be-captured documents to exhaust cropping resources is bad:

- Creates action at a distance, where JS libs can **DoS attack** cropping without user permission
- Defeating cropping may expose private user info in unsuspecting poorly-written apps 🦶🔫
- Early resource allocation inherently unnecessary; premature optimization to avoid cropTo IPC

DoS is avoided by simply doing IPC and resource allocation in cropTo. With that **baseline**, any earlier resource allocation is purely *UA optimization*, whose cost and complexity is borne by UA.

- **Precedent**: mediaSource.getHandle() is sync. Bigger Element & MessagePort don't exhaust
- **Privacy:** Exposing *global* resource limits to JS leaks cross-origin correlatable information
- Smaller *local* per-doc limits become exhaustible even by well-designed apps, web compat
- Web developers are unlikely to ever expect or check for exhaustion
- Better: UAs handle it, not fail (optimize for well-designed apps and fall back to baseline)

Issue 17: Why async CropTarget creation? (Jan-Ivar 5/6)

Performance: a part of the spec that doesn't have consensus says:

> "The user agent *MUST* resolve *p* only after it has finished all the necessary internal propagation of state associated with the new CropTarget, at which point the user agent *MUST* be ready to receive the new CropTarget as a valid parameter to cropTo."

Process separation of iframes means IPC of "state propagation" *AND* postMessage:

IPC 3 (POSTMSG)   IPC 2 (RSC ALLOC SUCCESS)   IPC 1 (STATE PROPAGATION)

```
// iframe.html
parent.postMessage(await CropTarget.fromElement(element), "*");
```

^ SERIALIZES 3 IPCs!

```
// toplevel.html
window.onmessage = async ({data: cropTarget}) => await track.cropTo(cropTarget);
```

Serializing IPC1 + IPC2 + IPC3 is slower than running IPC1 in parallel with IPC3.

cropTo can handle either approach. Proposed API is faster, simpler, still "optimizable":

```
parent.postMessage(new CropTarget(element), "*");
```

Satisfies §6.7 "The API implementation will not be blocked by … inter-process communication"

Issue 17: Why async CropTarget creation? (Jan-Ivar 6/6)

In conclusion (in support sync proposal, not meant to be neutral):

Early resource allocation is premature optimization, opens up DoS attacks on cropping

A sync API does not limit optimization. Burden is on UAs to optimize within the API (optimization-fail ≠ operation-fail. Implement a baseline in cropTo to fall back on)
New issues (#48) from optimization side effects not a good use of WG time. Let's be done.

There's no *inherent* need and no web developer benefit to it being async, and the async API goes against W3C design principles §2.4, §6.6, and §6.7.

There's also the web developer cost to it being async:

- Every `await` is a preemption point, requiring apps to reassess state (or risk data races)
- Much like non-const methods in c++, `async` propagates to all callers, and callers of callers etc.
- Having multiple processing failure points (for extremely rare resource errors) is risky
- It is slower (delaying when postMessage can happen)

CropTarget.fromElement Case for Async, Elad

Start Time: 07:35 AM

End Time: 07:50 AM

Region Capture API and Status

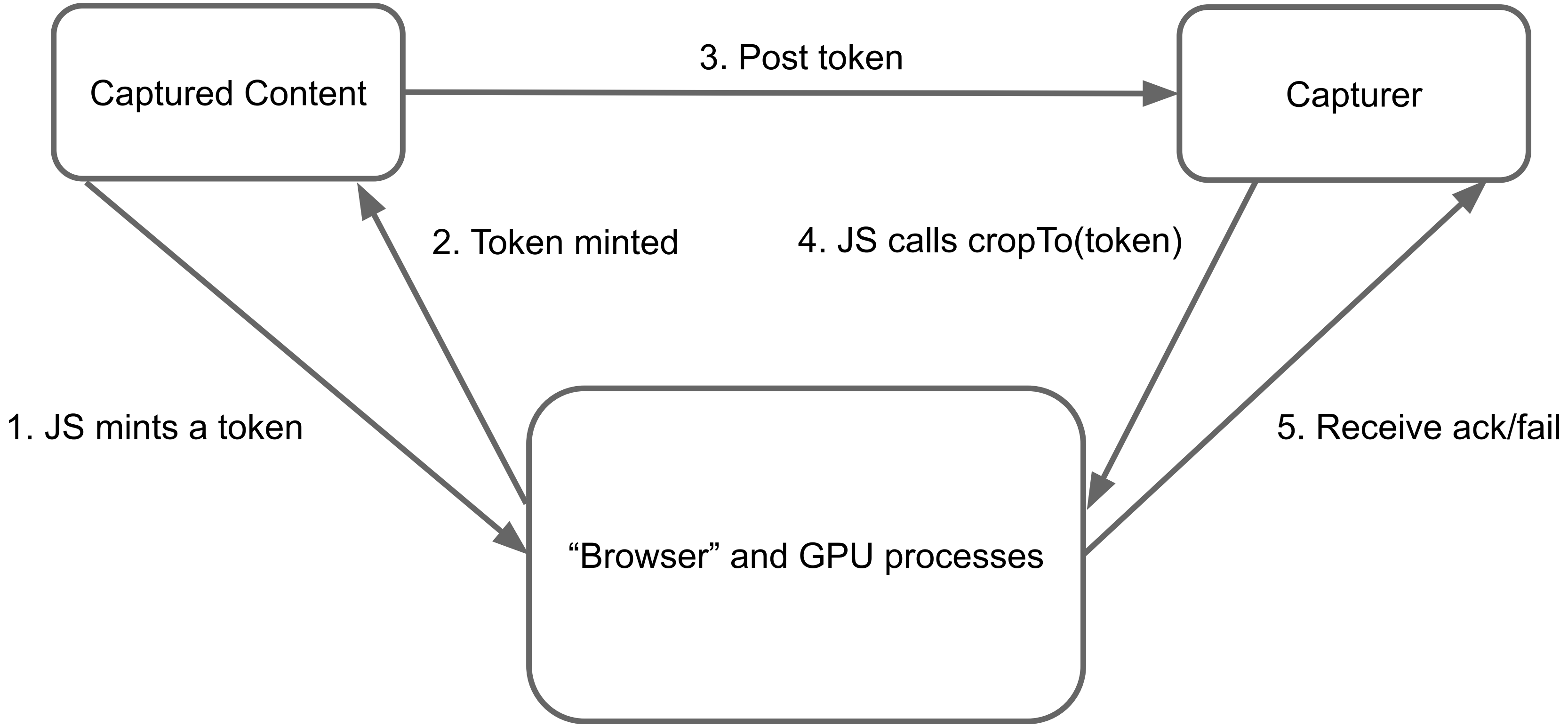
Region Capture:

- API for cropping video tracks.
- Proposed by your humble servant roughly 1.5 years ago.
- Implemented by Chromium.
- Successfully deployed by some of Web’s biggest apps. (1, 2)
- Battle-tested. (1)
- Chromium-team has implemented and learned their lessons.
- Mozilla and Apple have not implemented.

Recall Issue #17 (Token-minting asynchronicity)

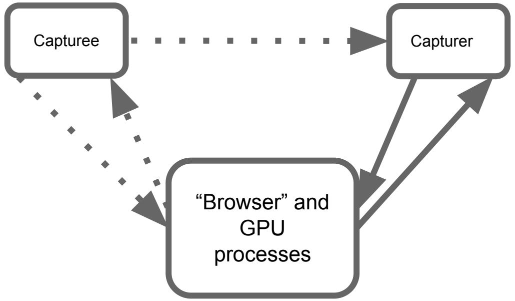
- Cropping involves a target.
- Targets are minted as CropTarget then passed to capturer.
- Is the CropTarget associated with the captured tab? Verification required.
- Verification only possible when calling cropTo(), as the capture may start before/after minting.
- When calling cropTo() - less IPCs => faster resolution.

Recall Chrome’s Implementation



Chrome’s Motivation for Own Implementation

- Simple and performant
  - cropTo() only makes one IPC round-trip.
  - This trip is to the “browser” process. Main-thread contention in the captured document does not slow down the capturer.
  - Token can even be minted ahead of time.
  - No race conditions.
- Flexible
  - Can change implementation later, because async is easier than sync.



Focal Point of Present Debate

Chrome **needs** CropTarget.fromElement() to be async.

- Our desired trade-offs set requires an async API.

Mozilla **prefers** CropTarget.fromElement() to be sync.

- Can trivially wrap sync result in a pre-resolved Promise.
- No change to the implementation and its set of trade-offs.

Unless Mozilla can demonstrate asynchronicity itself is an issue?
Let’s examine.

Priority of Constituencies

The W3C's Design Principles read:

- User needs come before the needs of web page authors,
- which come before the needs of user agent implementors,
- which come before than the needs of specification writers,
- which come before [redacted; discussed in upcoming slides].

(Source: https://w3ctag.github.io/design-principles/#priority-of-constituencies)

Cui bono? (Who profits?)

Users don’t know if the API is async. But Web-apps will be more performant and interoperable. Users will love that.

Web-developers don’t mind adding “await” in one line of their million-line application. They’ve told us as much. (1, 2)

Implementers demand this. The Chromium team, which is the ultimate authority on Chromium implementation, say it’s a Must. So for interoperability, this is required.

Arguments for Synchronicity

It’s been argued that all constituencies benefit from asynchronicity. What argument remains for synchronicity? Is it **theoretical purity**?

- TAG’s preference is inapplicable here, as IPCs require asynchronicity - according to TAG’s own principles.
- The W3C’s design principles expressly warn against dwelling on theoretical purity.
- TAG has explicitly criticized this Working Group for its inability to reach interoperability due to purity hang-ups. (Next slide.)

Theoretical Purity - W3C Design Principles

Remember the redacted slide? Here it is in full:

The W3C’s Design Principles read:

- User needs come before the needs of web page authors,
- which come before than the needs of user agent implementors,
- which come before than the needs of specification writers,
- which come before theoretical purity.

Theoretical purity is dead last.

TAG Feedback

Of the API as a whole:

- “we reviewed it and were **satisfied**”

Of the idea to make fromElement synchronous:

- Sangwhan: “I have looked at this discussion and think the developer ergonomic **gains would be minimal**… I don't see a significant gain in terms of ergonomics for developers with sync”
- Sangwhan: “it's **fine to be async**”
- Dan: “we can feed back that interoperability is an imperative”
- Dan: “[...] the issues of interop are many. This is something that should be guiding the WGs work. That should be the primary goal of anything. Not the most elegant or correct API”

Summary

- All constituents benefit from a flexible API that can be optimally implemented by each user agent according to their philosophy and their needs.
- Attempts to force a synchronous API serve no purpose other than to **inappropriately constrain implementations**.
- TAG has rightly criticized this Working Group for its inability to make progress and **prioritize interoperability**.
- Let’s do better. Much work awaits.

CropTarget.fromElement
Sync vs. Async - Shared Discussion

Start Time: 07:50 AM

End Time: 08:00 AM

CropTarget Name - Youenn
Start Time: 08:00 AM
End Time: 08:10 AM

Issue 18: Is CropTarget name too generic?

- CropTarget is made of two very generic terms
  - 'Crop'
    - Widely used term in the context of images
    - HTML mostly talks about masking & clipping
      - Using Crop seems fine
  - 'Target'
    - Does not bring much value (?)
- Can we find a more descriptive name?
  - Let's look at how to define CropTarget

Issue 18: Is CropTarget name too generic?

- What is a CropTarget?

  1. A serialized object used by APIs to refer to out-of-process DOM elements
  2. A reference to an out-of-process bounding box
  3. An object whose sole purpose is to be given to cropTo

- Why not a name that directly represents what this is?
  - ElementReference, BoundingBox
  - CaptureRegion, CropTarget, CropRegion

Issue 18: Is CropTarget name too generic?

- MediaStreamTrack.cropTo is dedicated to screen capture
    - We might want to crop other tracks in the future
        - Face detection based camera track cropping?
- It seems ok to favor the 3rd definition
    - An object whose sole purpose is to be given to cropTo

- How about CropRegion?
    - Bonus: it aligns with the spec name

Discussion

End Time: 08:10 AM

Region Capture Extensions (Elad)

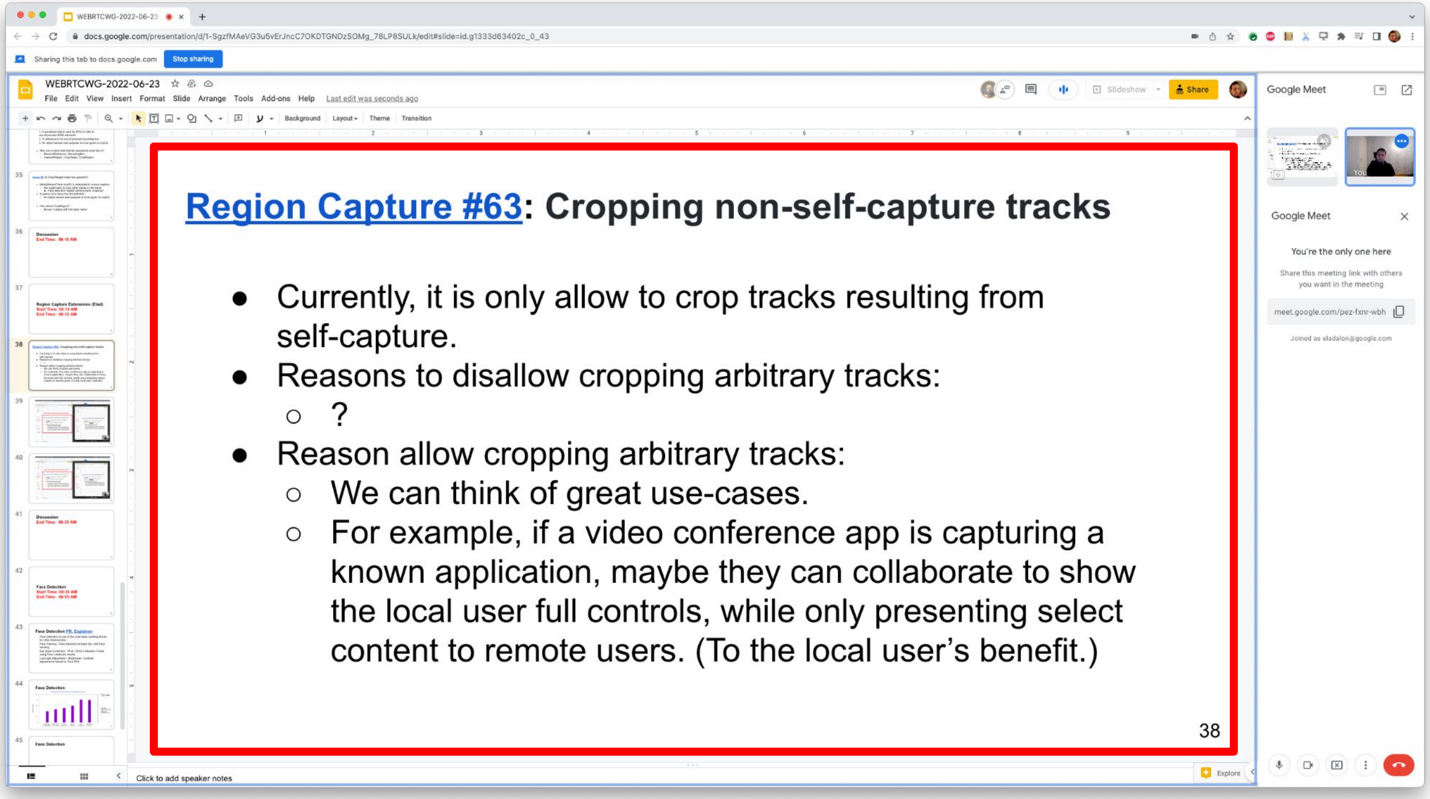
Start Time: 08:10 AM

End Time: 08:35 AM

Region Capture #63: Cropping non-self-capture tracks

- Currently, it is only allow to crop tracks resulting from self-capture.
- Reasons to disallow cropping arbitrary tracks:
  - ?
- Reason allow cropping arbitrary tracks:
  - We can think of great use-cases.
  - For example, if a video conference app is capturing a known application, maybe they can collaborate to show the local user full controls, while only presenting select content to remote users. (To the local user’s benefit.)

This is how I could do it today, using Region Capture over a self-capture track. The app could choose to only share the middle part.


WEBRTCWG-2022-06-23
docs.google.com/presentation/d/1-SgzfMAeVG3u5vErJncC7OKDTGNDzSOMg_78LP8SULk/edit#slide=id.g1333d63402c_0_43
Sharing this tab to docs.google.com
Stop sharing
File Edit View Insert Format Slide Arrange Tools Add-ons Help Last edit was seconds ago
Slideshow
Share
Background Layout Theme Transition
Google Meet
Click to add speaker notes
Explore

Google Meet

You're the only one here

Share this meeting link with others you want in the meeting

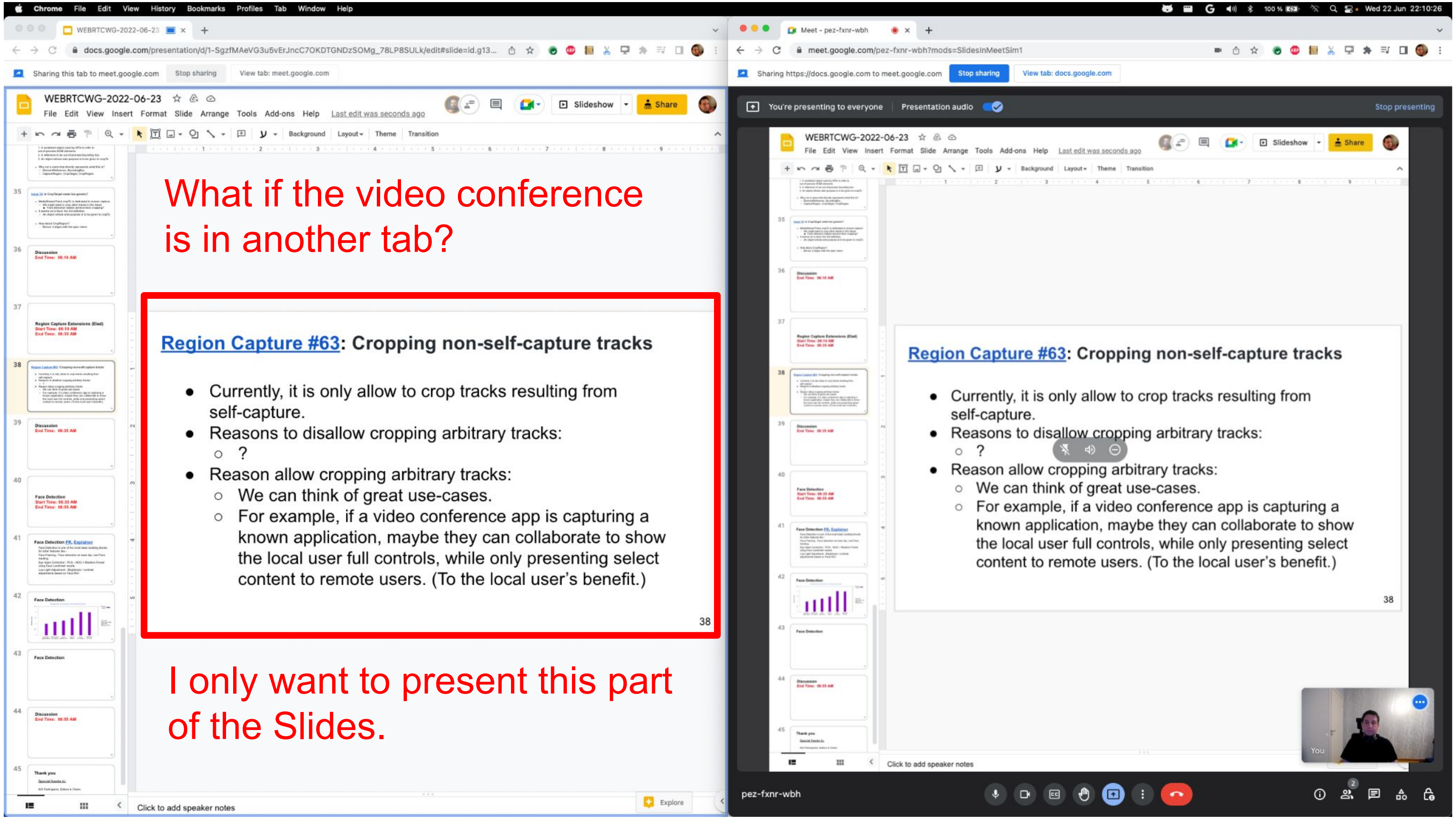
meet.google.com/pez-fxnr-wbh

Joined as eladalon@google.com

Region Capture #63: Cropping non-self-capture tracks

- Currently, it is only allow to crop tracks resulting from self-capture.
- Reasons to disallow cropping arbitrary tracks:
  - ?
- Reason allow cropping arbitrary tracks:
  - We can think of great use-cases.
  - For example, if a video conference app is capturing a known application, maybe they can collaborate to show the local user full controls, while only presenting select content to remote users. (To the local user's benefit.)

38



What if the video conference is in another tab?

Region Capture #63: Cropping non-self-capture tracks

- Currently, it is only allow to crop tracks resulting from self-capture.
- Reasons to disallow cropping arbitrary tracks:
  - ?
- Reason allow cropping arbitrary tracks:
  - We can think of great use-cases.
  - For example, if a video conference app is capturing a known application, maybe they can collaborate to show the local user full controls, while only presenting select content to remote users. (To the local user's benefit.)

38

I only want to present this part of the Slides.

Making CropTargets stringifiable

- Currently, CropTargets are serializable but not stringifiable. Why?
- Allowing stringification:
  - Allows communicating CropTargets over Capture Handle.
  - Allows communicating CropTargets over a shared cloud infrastructure between tabs which are cross-origin.
  - Allows utilization of existing string-based mechanisms for communication between documents, which some applications have. For example, when going over a shared cloud infrastructure.
- Avoiding stringification:
  - Apple has voiced concerns over persistence to storage, garbage collection, etc. Could these be restated so that we may examine the validity of the arguments?
  - Any other concerns?

Predictable Errors in Region Capture

- Web-developers hate API failures.
- Web-developers prefer APIs that never fail.
- But APIs fail for foreseeable and unforeseeable reasons.
  - Browser vendors choose to ship partial implementations.
  - Complete implementations are rendered incomplete by future changes in adjacent specs.
  - Confident engineers discover that challenges were greater than initially anticipated, and that their implementation does in fact run into various forms of resource-exhaustion.
- If an API fails when it shouldn't - are Web-developers comforted by the knowledge that the spec disallowed the failure?
- Will browser vendors refund the Web-developer for the non-compliant failure?
- Since failures are a fact of life, let's at least have consistent failures.

Discussion

End Time: 08:35 AM

Face Detection

Start Time: 08:35 AM

End Time: 08:55 AM

Face Detection PR, Explainer

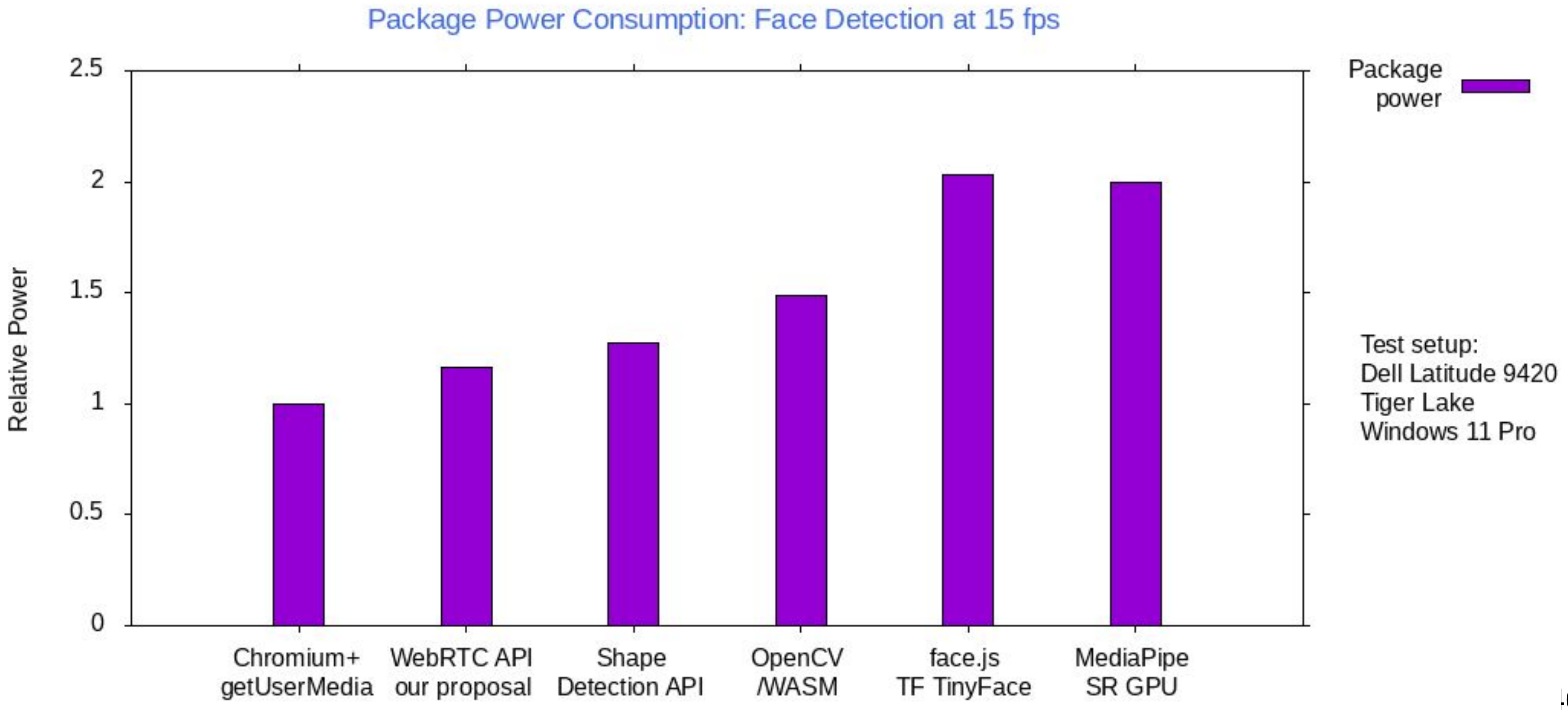
Face Detection is one of the basic building blocks for other features like -

Face Framing : Face detection at lower fps, rest Face tracking

Eye Gaze Correction : PCA - HOG + Random Forest using Face Landmark results.

Low Light Adjustment : Brightness / contrast adjustments based on Face ROI

Face Detection



Youenn’s comments

1. I would be tempted to make the API surface as minimal as possible (What is the MVP?) and leave the rest to a dedicated 'future steps' section. For instance, maybe the MVP only needs faceDetectionMode constraint (not landmarks/numfaces/contourpoints constraints) with a reduced set of values ("none" and "presence"). I am not sure about the difference between presence and contour for instance, which is somehow distracting. Are FaceLandmark part of the MVP as well?
2. The proposal is based on the VideoFrameMetadata construct, which is fine. We should try to finalise this discussion in WebCodecs.
3. DetectedFace has a required id and required probability. I can see 'id' being useful, maybe probability should be optional.

Discussion

End Time: 08:55 AM

Thank you

<ins>Special thanks to:</ins>

WG Participants, Editors & Chairs