W3C WebRTC WG TPAC Meetings

October 20 and 22, 2020
8 AM - 10:00 AM Pacific Time

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 virtual meetings of the W3C WebRTC WG at TPAC 2020!
- During these meetings, we hope to make progress on new work as well as bringing WG specifications to CR and PR.

About these Meetings

- Meeting info:
  - https://www.w3.org/2011/04/webrtc/wiki/TPAC_2020
- Link to latest drafts:
  - WebRTC 1.0 API: https://w3c.github.io/webrtc-pc/
  - WebRTC-Stats: https://w3c.github.io/webrtc-stats/
  - WebRTC-NV Use Cases: https://w3c.github.io/webrtc-nv-use-cases/
  - WebRTC Extensions: https://w3c.github.io/webrtc-extensions/
  - WebRTC-ICE: https://github.com/w3c/webrtc-ice
  - WebRTC SVC: https://github.com/w3c/webrtc-svc
  - Insertable Streams: https://github.com/w3c/webrtc-insertable-streams
  - WebRTC Priority: https://w3c.github.io/webrtc-priority/
  - WebRTC-DSCP: https://w3c.github.io/webrtc-dscp-exp/
  - Media Capture Automation: https://w3c.github.io/mediacapture-automation/
  - Media Capture and Streams: https://w3c.github.io/mediacapture-main/
  - Media Capture Image: https://w3c.github.io/mediacapture-image/
  - Media Capture Output: https://w3c.github.io/mediacapture-output/
  - Screen Capture: https://w3c.github.io/mediacapture-screen-share/
  - Audio output: https://w3c.github.io/mediacapture-output/
  - Media Recording: https://w3c.github.io/mediacapture-record/
  - Content-Hints: https://w3c.github.io/mst-content-hint/
- Link to Slides has been published on WG wiki
- Scribe? IRC http://irc.w3.org/ Channel: #webrtc
- The meeting is being recorded.

Agenda for Tuesday, October 20, 2020

8:00 AM - 8:15 AM State of the WebRTC WG (Harald)

8:15 AM - 8:45 AM Test and Implementation Status (Dr. Alex and Carine)

8:45 AM - 9:00 AM WebRTC-Stats (Henrik)
(https://w3c.github.io/webrtc-stats/)

9:00 - 9:30 AM Media Capture & Streams: Browser Picker Model (Jan-Ivar)

9:30 AM - 10:00 AM Other Capture specifications (Jan-Ivar and Youenn)

Agenda for Thursday, October 22, 2020

8:00 AM - 8:30 AM Insertable Streams (Raw Media) (Harald)

8:30 AM - 9:00 AM E2E Encryption (Youenn) (https://github.com/w3c/webrtc-insertable-streams/)

9:00 AM - 9:15 AM WebRTC-SVC (Bernard) (https://w3c.github.io/webrtc-svc/)

9:15 AM - 9:30 AM GetDisplayMedia and the Same Origin Policy - revisit (Jan-Ivar)

9:30 AM - 9:45 AM GetBrowserContextMedia proposal (eladalon) (https://github.com/w3c/mediacapture-screen-share/pull/148)

9:45 AM - 10:00 AM Wrapup and Next Steps (Chairs)

State of the W3C WEBRTC WG (Harald, 15 minutes)

What we're (re)chartered to do

- Finish WebRTC 1.0 (HIGH PRIORITY)
- Define an object-oriented API (based on ORTC)
- Describe requirements for new use cases
- Address those use cases
  - New protocols (and associated APIs)
  - New data access functions

What our environment demands

- WebRTC 1.0 should “just work”
  - Across all browsers
  - In all networks
- Low level data access
  - In a performant manner (example: link)
- Son of ORTC
  - Pressure seems to have decreased
  - WebTransport, WebCodec spinning out

Media Capture and Streams

- Candidate Recommendation (Oct 7, 2020)
- 27 open issues
  - 19 of which are > 3 months old
- Interoperability matrix shows lots of things working in ¾ of browsers
- Community sense seems to be “works”
- Promise: REC in Q1 2021

Screen Capture

- Push to bring functionality up to par with existing implementations based on gUM.
- Security still troublesome, but can’t live without
- TAG review needed

New development: getBrowserContextMedia proposal

WebRTC-1.0

- Candidate Recommendation
  - Recycled at CR Oct 15, 2020. Last CR prior to PR?
  - (famous last words)
- 10 open issues
  - 8 are > 3 months; test suite and editorials
- Interoperability matrix shows lots of issues, but also lots of interoperability
- Confluence map shows implementation progress from last year. But implementation plans seem to have slowed since the pandemic hit.
- Community sense “are we usable yet”?
- Promise: REC in Q4 2020 (Now)

WebRTC-Identity

- Candidate Recommendation (split sept 2018)
- 24 open issues
  - Newest issue is from 2017
- Test suite has been separated
- Old promise: PR in Q3 2019 (as for webrtc-pc)
- Community sense: “Not much happening”

Resources available to WG

- Editors: 2 editors (Henrik and Youenn) currently active on mediacapture-streams, screen-capture and webrtc-pc (Jan-Ivar continues to write)
  - Some others contribute PRs - THANK YOU!
- Other drafts managed by other editors

Where resources come from

- People are motivated to get stuff done that they care about
- Organizations sponsor people to get stuff done that they care about
- W3C is a “gift economy” - to make something happen, volunteer to work on it!
- Careful balance of “polish” vs “new work” needed - otherwise, new work goes elsewhere

Other documents - active

- Capture from DOM - heavy use
  - Need to find new editor(s)
  - Need privacy/security,TAG review
  - 19 open issues
- Recorder - heavy use, updates
  - Also one suggested path for “media access”
  - Need to find new editor(s)
  - Need privacy/security,TAG review
- Stats identifiers - updates
  - Linked to webrtc-pc

Other documents - quiet

- Depth - quieted down?
- Audio output devices - at CR, in use, little activity
- Content hints - released, PRs merged, only a few open issues.
- DSCP - no code, no activity
- WebRTC-SVC - “intent to implement” in Chrome
- WebRTC-ICE - implemented in Chrome

We should eventually kill or finish these.

Other W3C activity

- WebCodec WG - Media WG interest
- WebTransport WG - replacing RTP
- Timed Text - Web and TV
- Media Timed Events - Web and TV
- Media IG in general
- Security and Privacy issues

Attention focus for this meeting

- Finish 1.0
  - Get all the bugs resolved
  - Figure out how to get to interop across the board
- Look at new APIs
  - Where what we have is not enough
  - Use cases and requirements are key!
- Attend to Raw Media
  - Because that’s where we’re being asked to go

Testing and Implementation Status

(Carine Bournez and Dr. Alex Gouaillard, 30 minutes)

Testing ? by whom?

Issue 2412: Is the testing policy being applied/working? (Harald)

- Intent of testing policy
  - Whoever suggests a protocol change also supplies a test
  - Test suite is always in sync with spec, implementations trail
- Practice
  - Whoever implements a protocol change also makes a test
  - Spec leads, tests roughly follow fastest implementor
  - Not all aspects of protocol changes get tested
  - Some features have no WPT test coverage (e.g. multiplexing), others lack meaningful tests (e.g. degradationPreference), others do not address interop (simulcast)
  - Policy more focused on WPT than KITE tests.
  - Irregular review of WPT Issues and PRs:
    - 22 open WPT Issues, 10 more than a year old
    - 13 unmerged PRs, 5 more than 6 months old
- What can we do better?
  - Abandon testing policy?
  - Ask for resources for testing policy?
  - Adopt “no test - no merge”?
  - Once specification “final CR” issues, shift focus to review of test Issues and PRs?

WPT / Webrtc.org / webrtc github repos / discuss-webrtc

Merged **Fix/migrate to nightwatchjs** #1096
ErikHellman merged 49 commits into `gh-pages` from `fix/migrate-to-nightwatchjs` on Sep 4, 2018

ErikHellman commented on Sep 3, 2018 Contributor Author

One thing we should remember here is that these are samples showing how web developers how to do things. My intention is to show both how the API is used as well as how to do UI testing for WebRTC applications.

The tests here should not be part of the QA for WebRTC. It doesn't matter which browsers we're testing on since it is only for showing *how to test*. I'd prefer to have Chrome, Firefox AND Safari, but as there seems to be some problems with the geckodriver, I would be satisfied with Chrome and Safari.

fippo commented on Sep 3, 2018 • edited Contributor

But reducing the test matrix from

- chrome stable/beta/unstable
- firefox stable/beta/nightly

to

- chrome stable/beta
- safari stable

is counterproductive.

It was a long and bloody fight to get people to stop shipping webrtc stuff chrome-only (and I am not looking just at you hangouts). These samples (and the 'google recommends' that is implied) were important in delivering a "all browsers" message.

lgrahl commented on Sep 3, 2018 • edited Contributor

I share **@fippo**'s concern. If this is about showing people how to test WebRTC and if it doesn't work with Firefox at the moment, then people should stay away from it until it works. And so should the examples.

Perhaps we can nudge the people who are working on geckodriver regarding that particular issue. **@ErikHellman** is there a Bugzilla entry for the problem?

Testing from a non-browser member view

- Reminder:
  - Cosmo was one of the only three to volunteer to write tests originally before the W3C process required it
  - Cosmo wrote 1000 of the current 2000-ish test for the WG
- If you're a browser vendor, your test will be always accepted. Sometimes automatically merged from vendor repo to WPT repo.
- If you're not a browser vendor,
  - At best your test will need a reviewer/shepard. Alas, nobody has time, nor incentive to do that, see Issue #2412
  - More often than not, you will be ignored.
  - Then, your test will likely be challenged, perception being, not always for technical reasons,
  - Eventually, nobody will merge it.
  - Potential for major Issues to slip through the cracks. Example (mux/demux): https://bugs.chromium.org/p/chromium/issues/detail?id=1139052

Testing ? How much?

Spec Coverage Status - TPAC 2017

- PR #8051: Add coverage report and tools for WebRTC tests
- Coverage = (total - todo) / total
- Forcedly Obsoleted and not replaced by wpt group

```
$ cd webrtc/tools
$ node scripts/overview.js
Overall Coverage
====================
todo         |    248
tested       |    315
trivial      |    173
untestable   |     79
====================
total        |    815
coverage     | 69.57%
====================
```

4. Peer-to-peer connections	67.83%
5. RTP Media API	67.01%
6. Peer-to-peer Data API	71.87%
7. Peer-to-peer DTMF	93.54%
8. Statistics Model	100.00%
9. Identity	86.04%
10. Media Stream API Extensions for Network Use	35.71%

Spc Coverage Status - TPAC 2019/20 - ReSpec

- New Individual Project (ReSpec) by Dom, only links to WPT tests,
- Used by cosmo for AV1 RTP spec. Ext. for google unit tests and KITE tests
- No automated “Score”

Enumeration description	
closed ▶ tests: 1	The RTCPeerConnection object's [[IsClosed]] slot is true.
failed	t apply and any RTCIceTransports are in the "failed" state or any n the "failed" state.
disconne	es apply and any RTCIceTransports are in the "disconnected"
new ▶ tests: 1	None of the previous states apply and all RTCIceTransports are in the "new" or "closed" state, and all RTCDtlsTransports are in the "new" or "closed" state, or there are no transports.
connecting ▶ tests: 1	None of the previous states apply and any RTCIceTransport is in the "checking" state or any RTCDtlsTransport is in the "connecting" state.
connected ▶ tests: 1	None of the previous states apply and all RTCIceTransports are in the "connected", "completed" or "closed" state, and all RTCDtlsTransports are in the "connected" or "closed" state.

▼ tests: 1

- dtls-fingerprint-validation

The set of transports considered is the set of transports presently referenced by the PeerConnection's set of transceivers.

Decode Target Indication (DTI)

Describes the relationship of a frame to a Decode target. The DTI indicates four distinct relationships: ‘not present’, ‘discardable’, ‘switch’, and ‘required’.

Discardable indication

An indication for a frame, associated with a given Decode target, that it will not be a Referred frame for any frame belonging to that Decode target.

N ne Decode target may be discardable for one Decode target and not for an

▼ tests: 1

- scalability_structure_unittest

Fr

Describes frame dependency information for the coded video sequence. The structure includes the number of DTIs, an ordered list of Frame dependency templates, and a mapping between Chains and Decode targets.

Frame dependency template

Contains frame description information that many frames have in common. Includes values for spatial ID, temporal ID, DTIs, frame dependencies, and Chain information.

Not present indication

An indication for a frame, that it is not associated with a given Decode target.

▶ tests: 1

WPT

2018 ~ 2020 - WPT.fyi - interop mode

(3126 => 3497)

More tests, same distribution.

2020 a slow year.

COVID or lack of interest (or problems cited before)

	0	1	2	3	4
2018	32%	26%	31%	10%	0%
2019	5%	10%	23%	20%	42%
2020	4%	12%	20%	21%	44%

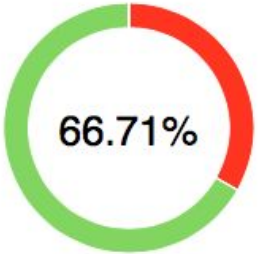
TPAC 2019 - WPT.kite (3126)

ALLURE REPORT 9/18/2019

17:25:49 - 18:40:45 (1h 14m)

18969

test cases



SUITES 7 items total

MAC_fi_69	1220	1906
WIN_fi_69	1184	1942
MAC_fi_71	1179	1947
WIN_fi_71	1176	1950
MAC_ch_77	775	2454
WIN_ch_77	773	2456

CATEGORIES 18 items total

Product defects	2175
Media Issues	1083
Stream Issues	969
Track Issues	957
State Issues	664
Configuration Issues	484
Parameters Issues	414
Candidate Issues	358
Codec Issues	338
Transceiver Issues	246

Show all

TPAC 2020 - WPT.kite- File Mode

Suites

order | name | duration | status

Status: 1408 0 1044 0 0 Marks:

- LIN_ch_86 (2020-10-19-165554) 160 164
- LIN_fi_81 (2020-10-19-174112) 193 131
- MAC_ch_86 (2020-10-19-165555) 160 164
- MAC_fi_81 (2020-10-19-173515) 193 131
- MAC_sa_14 (2020-10-19-182308) 226 98
  - media-capabilities (8/44) 3
  - media-feeds (1/2) 1
  - media-playback-quality (19/31) 1
  - media-source (402/563) 46 25
  - mediacapture-depth (11/11) 1
  - mediacapture-fromelement (42/55) 4 1
    - #1 capture.html (3/7) 5s 987ms
    - #2 creation.html (4/7) 5s 623ms
    - #3 ended.html (3/7) 7s 433ms
    - #4 historical.html (2/2) 455ms
    - #5 idlharness.window.html (30/32) 16s 216ms

2020-10-19-165553_http://wpt.live/mediacapture-fromelement/idlharn…

Failed **idlharness.window.html (30/32)**

Overview | History | Retries

```
assert_own_property: interface prototype object missing non-static
operation expected property "captureStream"
missinghttp://wpt.live/resources/idlharness.js:2474:32
```

Categories: Capture Issues Stream Issues Capture Issues Stream Issues Capture Issues Stream Issues Capture Issues Stream Issues Capture Issues Stream Issues Capture Issues Stream Issues Capture Issues Stream Issues Capture Issues Stream Issues

Severity: normal

Duration: 16s 216ms

Description

idlharness.window.html

Execution

Test body

- Harness status 16s 216ms
- idl_test setup 16s 216ms
- idl_test validation 16s 216ms
- Partial interface HTMLMediaElement: original interface defined 16s 216ms
- Partial interface HTMLMediaElement: member names are unique 16s 216ms

TPAC 2019 - WPT.kite- Individual Mode

order name duration status

Status: 8741 0 16387 0 0 Marks:

- Web Platform Test 8
- WIN_fi_69 1183
- MAC_fi_69 1221
- WIN_fi_71 1177
- MAC_fi_71 1160
- AND_ch_77 780
  - webrtc 301
    - #1520 RTCCertificate interface: operation getSupportedAlgorithms() 6m 46s
    - #1515 RTCErrorEvent interface: new RTCErrorEvent('error') must inherit property "error" with the proper type 6m 46s
    - #1508 RTCStatsEvent interface: existence and properties of interface object 6m 46s
    - #1494 RTCSessionDescription interface: attribute sdp 6m 46s
    - #1493 RTCPeerConnection interface: operation setLocalDescription(RTCSessionDescriptionInit, VoidFunction, RTCPeerConnectionErrorCallback) 6m 46s
    - #1466 RTCStatsEvent interface object length 6m 46s
    - #1462 RTCStatsEvent interface: attribute report 6m 46s
    - #1452 RTCErrorEvent must be primary interface of new RTCErrorEvent('error') 6m 46s

2019-09-17-201103_https://w3c-test.org/webrtc/idlharness.https.window..

Failed **RTCCertificate interface: operation getSupportedAlgorithms()**

Overview History Retries

```
RTCCertificate interface: operation getSupportedAlgorithms()
```

Categories: Certificate Issues Certificate Issues Certificate Issues Certificate Issues Certificate Issues Certificate Issues

Severity: normal

Duration: 6m 46s

Description

https://w3c-test.org/webrtc/idlharness.https.window.html

Execution

Test body

RTCCertificate interface: operation getSupportedAlgorithms() 1 attachment 6m 46s

Result 233 B


```
{"name":"RTCCertificate interface: operation getSupportedAlgorithms()","status":"FAIL","message":"assert_own_property: interface object missing static operation expected property \"getSupportedAlgorithms\" missing","expected":"PASS"}
```

TPAC 2020 - WPT.kite (3497)

ALLURE REPORT 10/20/2020

16:05:50 - 17:38:38 (1h 32m)

27095

test cases

SUITES 9 items total

MAC_sa_14 (2020-10-20-164716)	1439	2058
LIN_fi_81 (2020-10-20-160557)	1160	2379
MAC_fi_81 (2020-10-20-160557)	1158	2381
WIN_fi_81 (2020-10-20-160748)	1067	2472
WIN_ch_86 (2020-10-20-160554)	787	2853
LIN_ch_86 (2020-10-20-160550)	784	2858
MAC_ch_86 (2020-10-20-160551)	784	2856
MAC_Sa_14.1 (2020-10-20-164903)	777	1274

CATEGORIES 20 items total

Product defects	2475
Media Issues	1548
Track Issues	1241
Stream Issues	1120
Connection Issues	770
Configuration Issues	567
Parameters Issues	524
Codec Issues	498
Candidate Issues	495
State Issues	491

Show all

TREND

Simulcast (& SVC)

Simulcast Testing history

- From start of the WG we knew WPT can’t test p2p, need network for ICE, ...
- TPAC 2015 @ Sapporo: decision to include Simulcast in 1.0
  => can’t test SFU based scenarios (Simulcast) as needed by webrtc 1.0
- TPAC 2016 @ Lisbon: KITE Design Proposal
  => KITE implementation with google help
- TPAC 2018 @ Lyon: Simulcast frightening: cosmo pledge to help and provide tools
  => Simulcast / SVC tests implemented provided to all. Ref test, ref SFU.
- 2019 : multiple interactions with the WPT group, extend KITE to provide wpt.fyi compliant format, …. FAIL.
  => Move Simulcast testing to IETF hackathons, where most the webrtc browser guys show up a small group of motivated people exists (SFU developers).
  => IETF (104, 105, …) Hackathon !
- 2020: Simulcast loopback tests introduced. Does not address interop, but did find major bugs (e.g. lack of support for MID).

Status Report - Browser support card

			chrome 75 (canary)	chrome stable	Safari TP	Safari	firefox
Media Simulcast / ABR		h264 simulcast	yes - but bug pending	only via SDP mungling	yes	yes	no
		vp8 simulcast	yes	only via SDP mungling	yes	yes	yes
W3C Browsers APIs	RTCTransceiver	Have transceivers	yes - with unified plan	yes - unified plan	yes	yes	yes
	Stats API	Compliant Stats	yes - but bug pending	no	no	no	no
		Per layer Stats	no	no	no	no	no
	Simulcast enabling	Standard API + createOffer()	yes	no	no	no	yes - but old setParameter()
		legacy SDP mangling	yes	yes	yes	yes	no
IETF Internet protocols	Signalling (JSEP, SDP O/A)	Standard Unified Plan	yes	yes	yes	opt-in	yes
		Legacy Plan B	opt-in	opt-in	opt-in	yes	no
	Media Transport (RTP) simulcast features	rid	yes - if using addTransceiver	no	no	no	yes
		repairedId (RTX)	yes	no	no	no	no (no RTX at all)
		legacy ssrc in SDP	no - if using addTransceiver	yes	yes	yes	yes
	Bandwidth evaluation and congestion control	transport-wide-cc	yes	yes	yes	yes	no
		REMB	yes	yes	yes	yes	yes
	not all standards, but some IETF doc exists		vetted by henrik and harald		vetted by Youenn		vetted by nils

Status Report
Open WebRTC SFU support table

									Commercial PaaS
tested at IETF 104		Yes	Yes	Yes	No	No	No	No	No
team member present at IETF 104		Yes	Yes	No	No	No	No	No	No
Point of Contact		sergio	lorenzo	inaki	Voluntas	emil / boris / saul	?	micael gallego	gustavo garcia
Name		medooze	janus (VideoRoom plugin)	mediasoup	sora shiguredo	jitsi	licode	openvidu / KMS	houseparty
SDP Plan semantics	Unified Plan	yes	yes	yes	yes	One way only through convertion	no	no	no
simulcast enabled via	addTransceiver	yes	yes	yes	no	no	no	simulcast not supported	no
PC and stream handling	single multi-stream PC	yes	sending (MID and RID), receiving (SSRCs), both with BUNDLE ("unified-plan" branch)	sending (MID and RID), receiving (SSRCs), both with BUNDLE	yes	yes	it's flexible, depends on scalability: M multistream x N PC	no	yes
video codecs	VP8	yes + simulcast	yes + simulcast	yes + simulcast	yes + simulcast	Depends on configuration, but mainly VP8	yes + simulcast	yes	yes
	H.264	yes + simulcast	yes + simulcast	yes + simulcast	yes + simulcast		yes	yes	no
IDs	rids supported	yes	yes	yes	yes	no	yes	no	no
	repairid supported	yes	yes	yes	yes	no	no	no	no
	ssrc-less supported	yes	yes (simulcast only)	yes	no	no	no	no	no

Status Report
Open WebRTC SFU support table

		Open Source Media Servers								Commercial PaaS		
tested at IETF 104		Yes	Yes	No	Yes	No	No	No	No	No		
team member present at IETF 104		Yes	Yes	No	No	No	No	No	No	No		
Point of Contact		lorenzo	sergio	emil / boris / saul	inaki	?	?	micael gallego	Voluntas	gustavo garcia	?	?
Name		janus (VideoRoom plugin)	medooze	jitsi	mediasoup	INTEL	licode	openvidu / KMS	sora shiguredo	houseparty	tokbox	twilio
SDP Plan semantics	Plan B	yes	yes	Yes	yes		yes	yes	yes	yes		
	Unified Plan	yes	yes	One way only through convertion	yes		no	no	yes	no		
SDP O/A signaling	direct SDP signalling	yes	yes	no	ORTC: RTCRtpParameters on the wire, SDP O/A locally		yes	yes	yes	no		
	other	no	JSON on the wire, SDP locally	Jingle / COLIBRI on the wire, SDP locally			no	no	no	JSON on the wire SDP locally		
simulcast enabled via	SDP munging	yes	yes	yes	yes		yes	simulcast not supported	yes	yes		
	setParameter	yes	yes	no	yes		yes		no	no		
	addTransceiver	yes	yes	no	yes		no		no	no		
PC and stream handling	separate publisher and Subscriber PC	yes	no	no	yes.		no	yes	no	no		
	multiple PC	"master" => multiple,	no	no	no		it's flexible, depends on scalability: M multistream x N PC	?	no	no		
	single multi-stream PC	"unified-plan" => single multistream PC	yes	yes	sending (MID and RID), receiving (SSRCs), both with BUNDLE				yes	yes		
video codecs	VP8	yes + simulcast	yes + simulcast	Depends on configuration, but mainly VP8	yes + simulcast		yes + simulcast	yes	yes + simulcast	yes		
	H.264	yes + simulcast	yes + simulcast		yes + simulcast		yes	yes	yes + simulcast	no		
	VP9	yes + SVC	yes + SVC		yes		yes + SVC	no	no	no		
	rids supported	yes	yes	no	yes		yes	no	no	no		
	repairid supported	yes	yes	no	yes		no	no	yes	no		
	ssrc-less supported	yes (simulcast only)	yes	no	yes		no	no	no	no		
bandwidth congestion control	transport-wide-cc	yes - only receiver side	yes	yes	no		no	no	yes - only receiver side	yes		
	remb	yes	yes	yes	yes		yes	yes	yes			
bandwidth limitation on senders		REMB + SDP AS	no	simulcast layers dropping	proprietary client API		proprietary client API	Proprietary client API or settings	no	REMB		
mid rewritting		no	yes	no	no		no	no	yes	no		

Note on Simulcast Testing

- The IETF Hackathon **format** (browsers + SFU devs) seems to be highly effective.
- None of the open source SFU (except cosmo) is a W3C member. However they are all participating in IETF.
- It’s really tedious, and a lot of work, difficult to sustain in a gift economy. Because of COVID, CoSMo did not invest the time to organize and rally in 2020, and nothing happened.
- Some SFU vendors also start to propose competing webrtc Hackathon projects focussed on the protocols, and based at least in part on a rejection of the browser implementation and discuss-webrtc management.
- There are always 10 different IETF hackathons projects people would like to attend to, competing with this, even when not related to webrtc.
- In the absence of a reference test suite, there will be regressions.
- The stats are difficult to get right, even more difficult to test, and Henry is alone there. Help?

The W3C perspective on Testing

W3C Requirements for PR

- Process: https://www.w3.org/2020/Process-20200915/#transition-pr
- Criteria:
  - *must* show adequate implementation experience except where an exception is approved by the Director,
  - *must* show that the document has received wide review,
  - *must* show that all issues raised during the Candidate Recommendation review period other than by Advisory Committee representatives acting in their formal AC representative role have been formally addressed,
  - *must* identify any substantive issues raised since the close of the Candidate Recommendation review period,
  - *must not* have made any substantive changes to the document since the most recent Candidate Recommendation Snapshot, other than dropping features identified at risk.
  - *may* have removed features identified in the Candidate Recommendation Snapshot document as at risk without republishing the specification as a Candidate Recommendation Snapshot.

WebRTC 1.0 Issues

- Only 10 Open Issues. Labels:
  - Editorial: 4
  - Ready for PR (Pull Request): 7 (including 3 editorial issues)
  - Test suite issue: 4
  - Question: 1
- New issue velocity: 4 in the last month (2 editorial)
- Current fix velocity: 6 in the last month

More testing issues to come?

WPT/WebRTC Status

- WPT status: https://wpt.fyi/webrtc a lot more green boxes
- Annotated interoperability report: https://w3c.github.io/webrtc-interop-reports/webrtc-pc-report.html
  - Apart from a few mandatory stats, the only feature without any implementation (voiceActivityFlag) is already marked at risk
  - Only 1 implementation of: SctpTransport, IceTransport, setStreams, DataChannel.onclosing
- Report for simulcast
- IDL Interface tracker: https://dontcallmedom.github.io/webrtc-impl-tracker/?webrtc
  - Now matches the WPT test results: gaps in SctpTransport, DtlsTransport and IceTransport areas. Globally, implementations are a lot more interoperable.

TPAC 2019 -> TPAC 2020

Path	Chrome 78 Linux 18.04 1719e88 Sep 13, 2019	Edge 78 Windows 10.0 1719e88 Sep 13, 2019	Firefox 71 Linux 18.04 1719e88 Sep 13, 2019	Safari 82 preview macOS 10.13 1719e88 Sep 13, 2019
RTCCertificate-postMessage.html	3 / 4	3 / 4	1 / 4	4 / 4
RTCCertificate.html	5 / 6	5 / 6	2 / 6	5 / 6
RTCConfiguration-bundlePolicy.html	16 / 16	16 / 16	8 / 16	16 / 16
RTCConfiguration-iceCandidatePoolSize.html	10 / 10	10 / 10	1 / 10	10 / 10
RTCConfiguration-iceServers.html	33 / 76	33 / 76	30 / 76	32 / 76
RTCConfiguration-iceTransportPolicy.html	14 / 17	14 / 17	11 / 17	17 / 17
RTCConfiguration-rtcpMuxPolicy.html	14 / 14	14 / 14	1 / 14	14 / 14
RTCDTMFSender-insertDTMF.https.html	7 / 8	6 / 8	8 / 8	1 / 8
RTCDTMFSender-ontonechange-long.https.html	2 / 2	2 / 2	2 / 2	1 / 2
RTCDTMFSender-ontonechange.https.html	12 / 14	12 / 14	14 / 14	1 / 14
RTCDataChannel-bufferedAmount.html	11 / 13	11 / 13	13 / 13	1 / 13
RTCDataChannel-id.html	5 / 5	5 / 5	5 / 5	1 / 5
RTCDataChannel-send-blob-order.html	1 / 2	1 / 2	1 / 2	1 / 2
RTCDataChannel-send.html	7 / 12	7 / 12	11 / 12	8 / 12
RTCDataChannelEvent-constructor.html	5 / 5	5 / 5	5 / 5	5 / 5
RTCDtlsTransport-getRemoteCertificates.html	2 / 2	2 / 2	1 / 2	1 / 2
RTCDtlsTransport-state.html	4 / 4	4 / 4	1 / 4	1 / 4
RTCError.html	24 / 24	24 / 24	1 / 24	1 / 24
RTCIceCandidate-constructor.html	19 / 19	19 / 19	17 / 19	8 / 19
RTCIceConnectionState-candidate-pair.https.html	2 / 2	1 / 2	2 / 2	2 / 2
RTCIceTransport-extension.https.html	29 / 30	30 / 30	1 / 30	1 / 30
RTCIceTransport.html	1 / 3	1 / 3	1 / 3	1 / 3
RTCPeerConnection-add-track-no-deadlock.https.html	2 / 2	2 / 2	2 / 2	2 / 2
RTCPeerConnection-addIceCandidate.html	17 / 30	17 / 30	30 / 30	11 / 30
RTCPeerConnection-addTrack.https.html	10 / 10	6 / 10	10 / 10	10 / 10
RTCPeerConnection-addTransceiver.https.html	11 / 13	9 / 13	11 / 13	11 / 13
RTCPeerConnection-canTrickleIceCandidates.html	1 / 4	1 / 4	4 / 4	1 / 4
RTCPeerConnection-connectionState.https.html	7 / 7	5 / 7	1 / 7	5 / 7
RTCPeerConnection-constructor.html	22 / 23	22 / 23	21 / 23	22 / 23
RTCPeerConnection-createAnswer.html	3 / 4	3 / 4	4 / 4	4 / 4
RTCPeerConnection-createDataChannel.html	37 / 42	37 / 42	35 / 42	26 / 42
RTCPeerConnection-createOffer.html	3 / 5	3 / 5	5 / 5	4 / 5
RTCPeerConnection-generateCertificate.html	7 / 9	7 / 9	9 / 9	9 / 9
RTCPeerConnection-getDefaultIceServers.html	1 / 2	1 / 2	1 / 2	1 / 2
RTCPeerConnection-getStats.https.html	9 / 14	8 / 14	8 / 14	6 / 14
RTCPeerConnection-getTransceivers.html	2 / 2	2 / 2	2 / 2	2 / 2
RTCPeerConnection-iceConnectionState-disconnected.https.html	2 / 2	1 / 2	2 / 2	2 / 2
RTCPeerConnection-iceConnectionState.https.html	12 / 12	7 / 12	8 / 12	7 / 12
RTCPeerConnection-iceGatheringState.html	3 / 4	3 / 4	3 / 4	3 / 4
RTCPeerConnection-mandatory-getStats.https.html	49 / 77	49 / 77	28 / 77	0 / 77
RTCPeerConnection-ondatachannel.html	5 / 9	5 / 9	7 / 9	4 / 9
RTCPeerConnection-onicecandidateerror.https.html	2 / 2	2 / 2	0 / 2	0 / 2

Path	Chrome 88 Linux 20.04 83cff49 Oct 18, 2020	Edge 87 Windows 10.0 83cff49 Oct 18, 2020	Firefox 83 Linux 20.04 83cff49 Oct 18, 2020	Safari 114 preview macOS 10.15 83cff49 Oct 18, 2020
RTCCertificate-postMessage.html	3 / 4	3 / 4	1 / 4	4 / 4
RTCCertificate.html	5 / 6	5 / 6	2 / 6	5 / 6
RTCConfiguration-bundlePolicy.html	16 / 16	16 / 16	8 / 16	16 / 16
RTCConfiguration-iceCandidatePoolSize.html	10 / 10	10 / 10	1 / 10	10 / 10
RTCConfiguration-iceServers.html	33 / 68	33 / 68	30 / 68	32 / 68
RTCConfiguration-iceTransportPolicy.html	14 / 17	14 / 17	11 / 17	17 / 17
RTCConfiguration-rtcpMuxPolicy.html	14 / 14	14 / 14	1 / 14	14 / 14
RTCDTMFSender-insertDTMF.https.html	8 / 8	8 / 8	8 / 8	8 / 8
RTCDTMFSender-ontonechange-long.https.html	2 / 2	2 / 2	2 / 2	2 / 2
RTCDTMFSender-ontonechange.https.html	14 / 15	14 / 15	15 / 15	15 / 15
RTCDataChannel-binaryType.window.html	2 / 8	2 / 8	3 / 8	3 / 8
RTCDataChannel-bufferedAmount.html	21 / 25	21 / 25	25 / 25	25 / 25
RTCDataChannel-close.html	9 / 9	9 / 9	3 / 9	0 / 1
RTCDataChannel-id.html	5 / 5	5 / 5	5 / 5	1 / 5
RTCDataChannel-send-blob-order.html	1 / 3	1 / 3	1 / 3	3 / 3
RTCDataChannel-send.html	12 / 22	12 / 22	20 / 22	18 / 22
RTCDataChannelEvent-constructor.html	5 / 5	5 / 5	5 / 5	5 / 5
RTCDtlsTransport-getRemoteCertificates.html	2 / 2	2 / 2	1 / 2	1 / 2
RTCDtlsTransport-state.html	4 / 4	4 / 4	4 / 4	1 / 4
RTCError.html	24 / 24	24 / 24	1 / 24	1 / 24
RTCIceCandidate-constructor.html	19 / 19	19 / 19	17 / 19	19 / 19
RTCIceConnectionState-candidate-pair.https.html	2 / 2	2 / 2	2 / 2	2 / 2
RTCIceTransport-extension.https.html	32 / 32	32 / 32	1 / 32	1 / 32
RTCIceTransport.html	1 / 3	1 / 3	1 / 3	1 / 3
RTCPeerConnection-SLD-SRD-timing.https.html	2 / 2	2 / 2	2 / 2	1 / 2
RTCPeerConnection-add-track-no-deadlock.https.html	2 / 2	2 / 2	2 / 2	2 / 2
RTCPeerConnection-addIceCandidate-connectionSetup.html	4 / 4	4 / 4	4 / 4	1 / 4
RTCPeerConnection-addIceCandidate-timing.https.html	5 / 5	5 / 5	5 / 5	5 / 5
RTCPeerConnection-addIceCandidate.html	17 / 30	17 / 30	30 / 30	10 / 30
RTCPeerConnection-addTrack.https.html	10 / 10	9 / 10	10 / 10	10 / 10
RTCPeerConnection-addTransceiver.https.html	13 / 13	13 / 13	11 / 13	11 / 13
RTCPeerConnection-canTrickleIceCandidates.html	4 / 4	4 / 4	4 / 4	1 / 4
RTCPeerConnection-candidate-in-sdp.https.html	2 / 2	2 / 2	2 / 2	2 / 2
RTCPeerConnection-connectionState.https.html	8 / 8	8 / 8	1 / 8	6 / 8
RTCPeerConnection-constructor.html	23 / 23	23 / 23	21 / 23	22 / 23
RTCPeerConnection-createAnswer.html	3 / 4	3 / 4	4 / 4	4 / 4
RTCPeerConnection-createDataChannel.html	47 / 51	47 / 51	51 / 51	33 / 51
RTCPeerConnection-createOffer.html	3 / 6	3 / 6	6 / 6	4 / 6
RTCPeerConnection-description-attributes-timing.https.html	5 / 5	5 / 5	5 / 5	1 / 5
RTCPeerConnection-explicit-rollback-iceGatheringState.html	4 / 4	4 / 4	4 / 4	0 / 4
RTCPeerConnection-generateCertificate.html	7 / 9	7 / 9	9 / 9	9 / 9
RTCPeerConnection-getStats.https.html	9 / 14	9 / 14	9 / 14	6 / 14
RTCPeerConnection-getTransceivers.html	2 / 2	2 / 2	2 / 2	2 / 2
RTCPeerConnection-helper-test.html	2 / 2	2 / 2	2 / 2	1 / 2
RTCPeerConnection-iceConnectionState-disconnected.https.html	2 / 2	2 / 2	2 / 2	2 / 2
RTCPeerConnection-iceConnectionState.https.html	13 / 13	13 / 13	9 / 13	8 / 13
RTCPeerConnection-iceGatheringState.html	8 / 8	8 / 8	6 / 8	2 / 8

WG discussion

- Features without double implementation: IceTransport, SctpTransport
- Based on implementers input, we are confident that we can get 2+ interoperable implementations of those, but not in a well-defined timeframe.
- Process 2020 allows to modify a Recommendation to correct normative bugs
  - Need to explicitly announce our intent to accept normative changes in Rec
- Proposal: do not delay first edition of WebRTC 1.0 Recommendation for longer, until more features have double interoperable implementation.

WebRTC-Stats
(Henrik, 15 minutes)

Updates since last TPAC

The focus of last year was primarily to enable simulcast stats by moving metrics around to appropriate dictionaries.

- The old stats hierarchy did not work for simulcast.
  - “track” stats was a mix of everything:
    - MediaStreamTrack metrics.
    - Sender/receiver metrics.
    - Encoding/decoding metrics.
  - “outbound-rtp” was not per-layer:
    - 3 simulcast layers at 30 fps showed up as a single “outbound-rtp” at 90 fps.

Updates since last TPAC

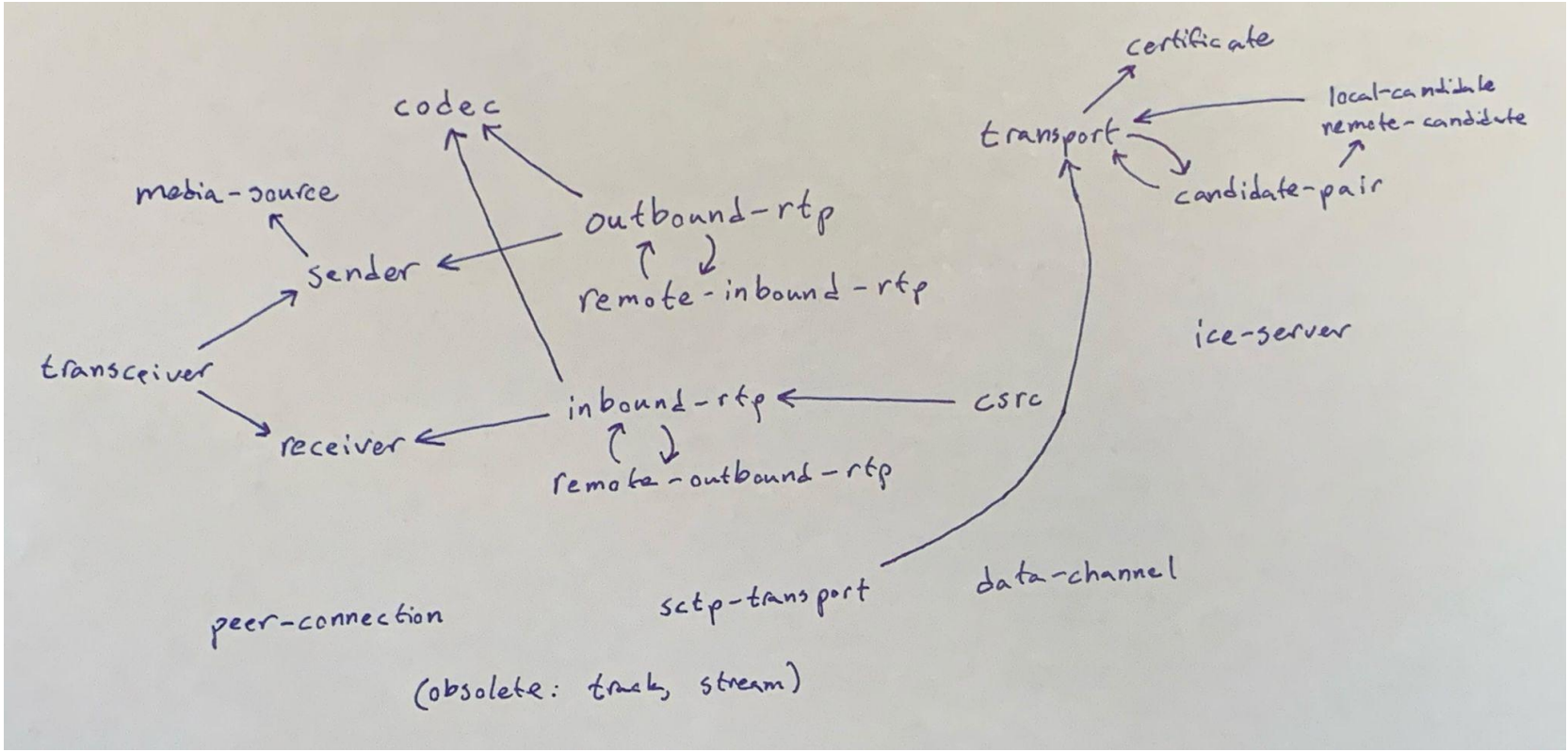
Simulcast stats migration has completed.

- “outbound-rtp” and “inbound-rtp” stats objects now contain encoding/decoding metrics previously found in “track” stats.
  - One “outbound-rtp” per simulcast layer.
- “media-source” now contain MediaStreamTrack-related metrics previously only found in “track” stats.
- “track” stats have moved to the obsolete section.

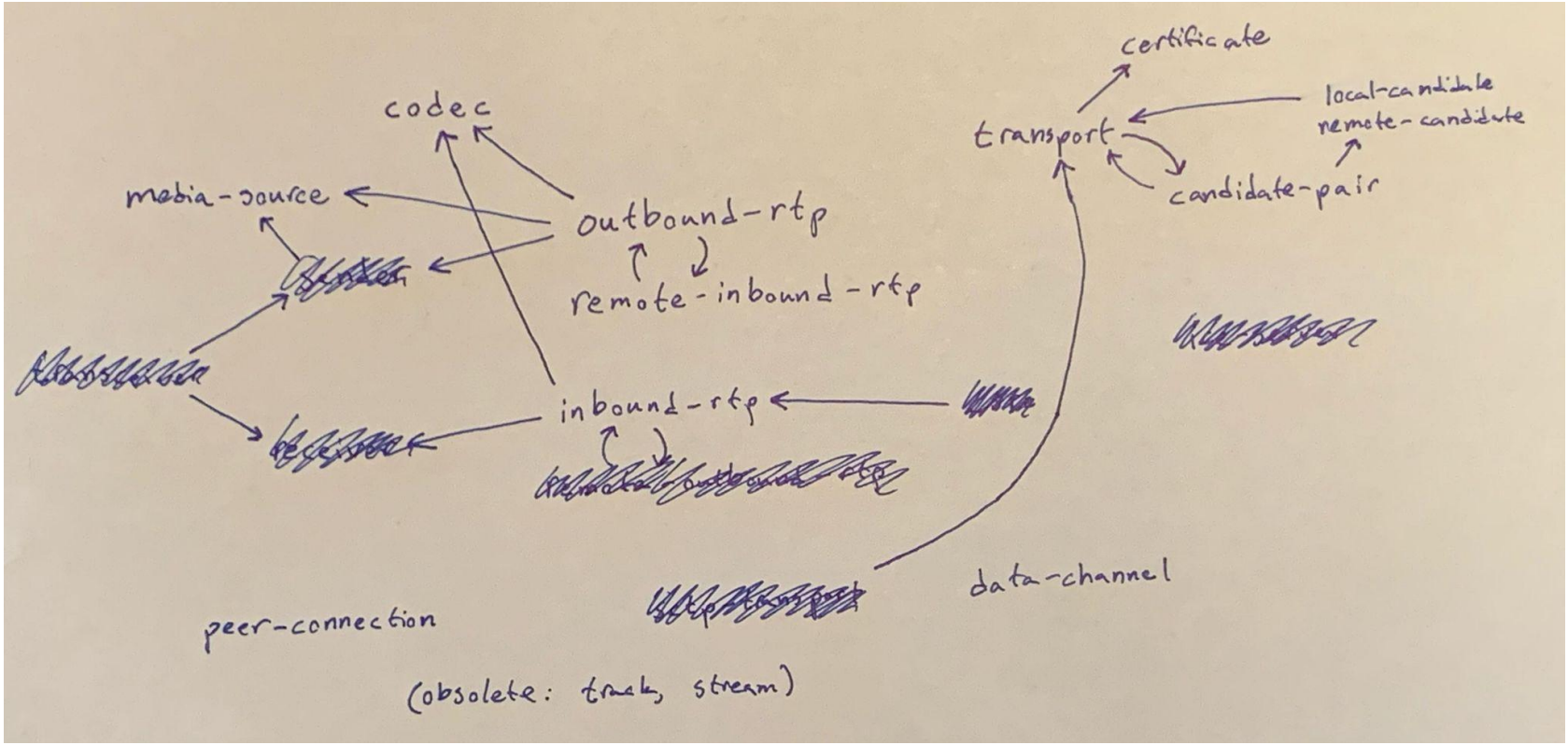
The migration completed in M86 (simulcast since M84).

- “track” stats still returned for backwards-compatibility reasons.

Spec Today

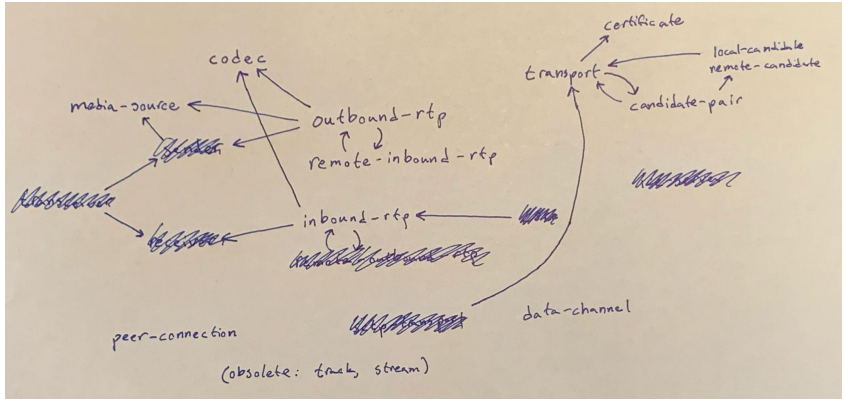

certificate
local-candidate
remote-candidate
codec
transport
media-source
outbound-rtp
candidate-pair
sender
remote-inbound-rtp
ice-server
transceiver
inbound-rtp
csrc
receiver
remote-outbound-rtp
data-channel
sctp-transport
peer-connection
(obsolete: track, stream)

Implementation Today (Chrome M86)



Implementation Today (Chrome M86)

What's missing?



- "remote-outbound-rtp" side of RTCP metrics.
  - Useful for one-way delay calculations with webrtc-extension's senderCaptureTimeOffset.
- Sender, receiver and transceivers.
  - Shows the relationship between these objects and the "transceiver.mid", but no new metrics per-se.
- SCTP transport metrics.
- ICE server metrics.
- CSRC -- already available outside of getStats():
  - getSynchronizationSources(), getContributingSources().

Implementation Status of Metrics

Mandatory stats

- WPTs show that 66/77 mandatory stats have been implemented in Chrome+Edge M87 (dashboard).

All stats

- X% implemented. *(Update slide when https://webrtc-stats.callstats.io/verify/ works.)*
- >= 170 metrics implemented.

Not much activity lately…

Very little activity on the spec lately.
Very little activity on the implementation-side, except for the stats migration.

- Does this mean that everything is Good Enough, or that we don’t have enough resources?

All-in-all, stats are in a pretty good shape.
Mostly “polishing” needed.

Media Capture & Streams (Jan-Ivar, 30 minutes)

State of privacy in Media Capture and Streams

Thanks to PING for reviewing these APIs!

```
await navigator.mediaDevices.enumerateDevices() // device enumeration
await navigator.mediaDevices.getUserMedia()     // camera/mic access
navigator.mediaDevices.ondevicechange = func    // detect device add/rem
```

12 issues were filed (4 open, 8 closed). 7 PRs were merged from review.

Media Capture & Streams (Jan-Ivar)

- **Privacy Issues** (Jan-Ivar)
  - #640 - Only reveal labels of devices user has given permission to
  - #645 - enumerateDevices should only provide info for granted device types
  - #646 - Should enumerateDevices by default return an empty list?
  - #672 - Deprecate inputDeviceInfo.getCapabilities() for better privacy
- **Other Issues**
  - #554: Specify webdriver way add/remove/setup web capture devices (Youennf)
  - #565: Should devicechange event fired when list of devices same? (Youennf)
  - #608: Is enumerateDevices list order significant? (Youennf)
  - #584 / PR 623: Resize mode (crop-and-scale) is under-specified (Henrik)
  - #660: Handling of rotation for camera capture streams (Jan-Ivar)
  - #735 / - Make fitness s/MAY/SHOULD/ for device selection (Jan-Ivar)

#640 - Only reveal labels of devices user has given permission to

Labels are bad for web compat and privacy, but will take time to get rid of.

Exposure significantly reduced now that a document must be capturing or have actively captured just now to see labels (persistent permission no longer sufficient).

Labels of non-granted devices are needed during capture to support sites implementing ⚙️ device pickers in browsers that don't grant all devices at once.

Long-term solution: in-browser picker for camera & mic (extension spec)

Short-term sub-issues:

1. Labels may contain private information. Encourage sanitization.
2. Clarify label is for display purposes; don't rely on == model/manufacturer.

Propose: Close issue after short-term solved, and revisit with in-browser picker extension spec.

#645 - enumerateDevices should only give info for granted types

Current spec allows enumeration of cams AND mics upon capturing either a camera OR microphone.

In theory it seems logical to further restrict this to allow

- enumerating cameras only if document is capturing/has captured camera
- enumerating microphones only if document is capturing/has captured mics

OTOH: Is successfully obtaining camera or microphone from the user perhaps sufficient to build a device picker for both?

Permission escalation example: Site X allows users to join web conferences with only microphone permission. Users expect to see camera choices in the site's ⚙️ options panel. Restriction may complicate ⚙️ UX, so site X demands camera on entry instead.

Consensus: Restrict devices by granted types. This is what Chrome is implementing so breakage risk is probably low.

#646 - Should enumerateDevices by default return an empty list?

Not web compatible to return an empty list. Booleans enable cam/mic UX display.

Our thinking: allow user agents to fake camera and/or microphone if missing.

Side-effects: (inherent from loss of information; regardless of approach)

1. camera/mic related UX (buttons) always visible on sites that today hide them for users without camera and/or mic. Site only learns of absence when `getUserMedia` fails with `NotFoundError`. (Mild)
2. devicechange event will never fire when users plug in their first camera or mic. Site cannot take action on these events. (workflow issue?)

Consensus: Return booleans for cam/mic. The spec already allows user agents to fake devices (Safari has option to expose fake devices). **Propose:** Note this

#672 - Deprecate inputDeviceInfo.getCapabilities(); better privacy

This API helps sites enforce their constraints while building their pickers:

```
for (const device of await navigator.mediaDevices.enumerateDevices()) {
  if (device.getCapabilities().height.max < 1080) continue;
  options.push({name: device.label});
}
```

- But it lets site enumerate capabilities (min/max ranges, enums) of *all* devices.
- Only available during capture
- One implementation

Long term: A constraints-based in-browser picker would obsolete this need.

Side-effect of losing: User would be able to pick device violating site constraints.

No consensus. Feature at risk (1 implementation). Revisit w/in-browser picker

Issue 584/PR 623: Resize mode (crop-and-scale) is under-specified (Henrik)
OLD SLIDE SKIPPED

Problem: VideoResizeModeEnum's "crop-and-scale" is underspecified:

- **"*This resolution is downscaled and/or cropped from a higher camera resolution by the user agent.*"**

We don't want to allow stretching or black borders. The final media should be a subset of the input.

Proposal:

- **Add: "*The media MUST NOT be upscaled, stretched or have fake data created that did not occur in the input source.*"**

#660 - Handling of rotation for camera capture streams (Jan-Ivar)

What happens when phone is in portrait? https://jsfiddle.net/jib1/e5dkao41

```
getUserMedia({video: true});                    // getSettings() = 640x480
getUserMedia({video: {width: {min: 641}}}); // getSettings() = 720x480
                                                // getSettings() = 480x720
getUserMedia({video: {width: {min: 641}}}); // getSettings() = 480x720
```

Only getSettings() is rotated, not constraints, capabilities or constrainable props.

Proposal: Specify this, because… it's simple & what browsers are doing. 🤷🏻

#735 - Make fitness s/MAY/SHOULD/ for device selection (Jan-Ivar)

Right now selectSettings() is SHOULD but device selection's use of it is MAY (!)

Need better web compat around device selection. Important for this & other specs.

```
getUserMedia({video: {height: 1080, frameRate: 60});  // Which device is more
getUserMedia({video: {height: 1080, deviceId: last}); // important to the app?
getUserMedia({video: {height: 1080, zoom: 4}); // To zoom or not to zoom?
```

Millions of corner cases. Predictability trumps usefulness at the edges.
Let's just make it work the same across browsers.

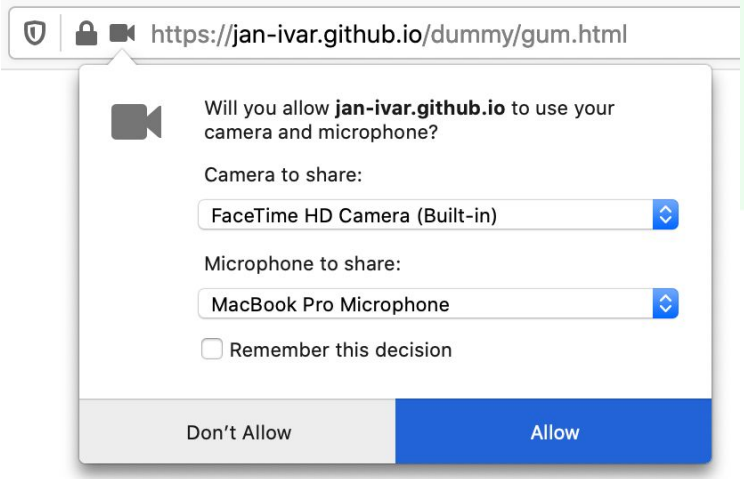
Key to fixing spec bugs (hacky `true` overloads) & avoid hand-waving in imageCapture

```
getUserMedia({video: {height: 1080, zoom: true}); // zoom loses (bug)
getUserMedia({video: {height: 1080, zoom: {}});   // ...because it's same as this
getUserMedia({video: {height: 1080});             // ...which is the same as this
```

#735 / PR 736 - Make fitness s/MAY/SHOULD/ for device selection

One exception: The user.

```
Once selected, the source of the
{{MediaStreamTrack}} MUST NOT change.</p>
-  <p>The User Agent MAY use the value of the computed
-  "fitness distance" from the <a>SelectSettings</a>
-  algorithm, or any other internally-available information
-  about the devices, as an input to the selection
-  algorithm.</p>
+  <p>The User Agent SHOULD use the value of the computed
   <a>fitness distance</a> from the <a>SelectSettings</a>
   algorithm as an input to the selection algorithm.
   However, it MAY also use other internally-available
   information about the devices, such as user preference.</p>
```



In-Browser picker moved to mediacapture-extensions

Long term we want to get away from in-content device selection

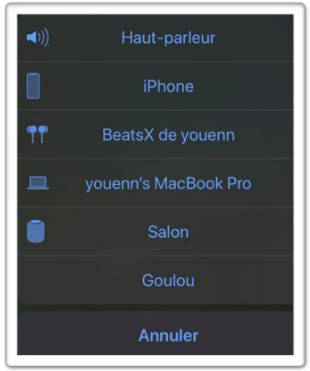
PING wants privacy-by-default *in-browser device picker*:

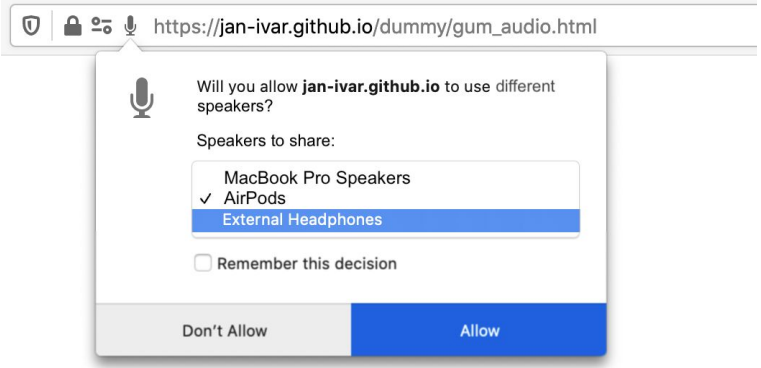
1. site asks for category (or categories) of device
2. *browser* prompts user for one, many or all devices
3. site gains access to only the device + label, of hardware the *user selects*.

State of speaker selection (in-browser picker)

Great progress in Audio Output API:

```
const id = await navigator.mediaDevices.selectAudioOutput(); // picker
audioElement.setSinkId(id); // redirect audio from default speakers
```





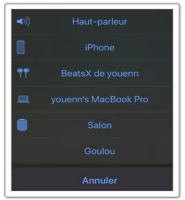
- Single id exposed in session in enumerateDevices() only *after* user picks.
- Works without microphone permission; redirect audio from any source.
- Off in iframes by default. Needs `allow="speaker-selection"`
- Firefox plans to implement soon. Thanks to Safari for driving design!

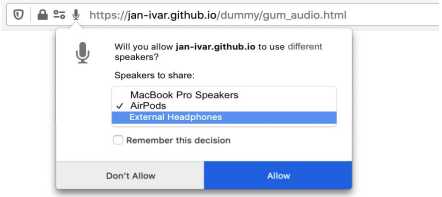
State of speaker selection (choice persistence)

Sites still need a way to remember device to not prompt every time (if user permits), but must call `selectAudioOutput` again to validate the id:

```
const deviceId = localStorage.speakerId; // from last visit
const id = await navigator.mediaDevices.selectAudioOutput({deviceId});
await audioElement.setSinkId(id);
localStorage.speakerId = id; // store id for next time (might be new)
```

If accepted, the picker is skipped. But the user agent may show picker at times (e.g. if the speaker device is no longer available), deterring trackers.





The id only appears in enumerateDevices if call succeeds.

In-Browser Cam/Mic Picker

So why not selectCamera() and selectMicrophone()? It's complicated:

- Web apps want constraints on camera selection (e.g. resolution)
- Web apps want some discovery (emerging use cases, streamers use 2 cams, WebVR)
- Users want sites to remember their configuration(s) and not pick device every time
- User agents differentiate in permission models (persistent on/off vs one-shot, innovation)
- **What would the migration path be?**
    - getUserMedia (unlike setSinkId) is already implemented in all browsers
    - Which sites will upgrade to prefer a less powerful / less established API?

Current Goal:

1. **Get rid of labels & capabilities of non-captured devices (consensus)**
2. ~~Prevent User agents from granting permission to all cameras/mics (no consensus)~~
3. ~~Limit capabilities exposure of in-use cam/mic ("availability API") (no consensus)~~

In-Browser Camera/Mic Picker Model (goals)

Incremental instead of new API

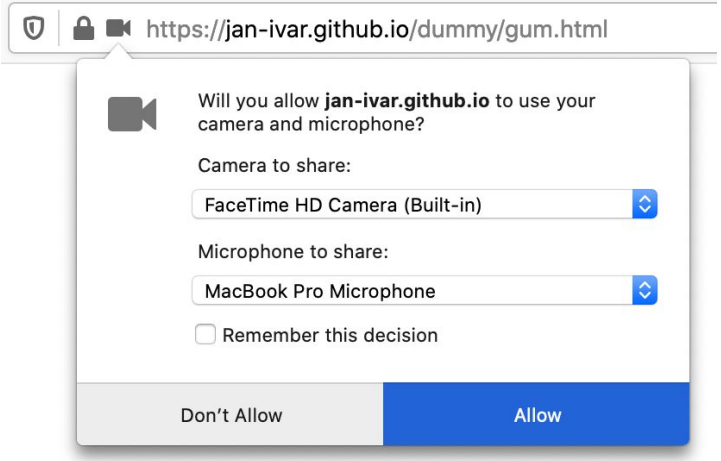
Device enumeration API (most powerful)	Label-less Device picker-style API	TAG/PING-definition picker-style API
getUserMedia() enumerateDevices()	**getUserMedia()++** **enumerateDevices()--**	*no "selectCamera()"* *no "selectMicrophone()"*
<u>During capture:</u> **All labels** All deviceIds **All** capabilities	<u>During capture:</u> All deviceIds Select capabilities	<u>During capture:</u> Select deviceIds Select capabilities
Implemented in all browsers	No implementation	No consensus on cam/mic Wouldn't get rid of getUserMedia or labels May revisit

LEAST POWERFUL

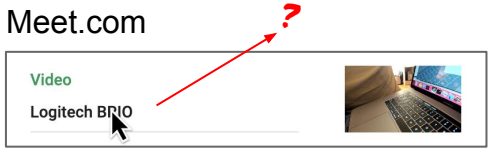
MIGRATION PATH

Incremental API

getUserMedia already has a picker in Firefox (tied to permission), letting the *user* instead of *user agent* choose within the app's constraints when choices >1



Meet.com



←Apps could have just called getUserMedia() again to get a different camera, but web compat prevents showing a prompt then, because lazy sites expect the same result (no prompt)

Incremental API (getUserMedia++)

Solution: Migrate to new getUserMedia semantics over time:

```
< await navigator.mediaDevices.getUserMedia({video: true, semantics: "browser-chooses"});
> await navigator.mediaDevices.getUserMedia({video: true, semantics: "user-chooses"});
```

New semantics mandate a picker if app constraints don't narrow down selection to 1 device per kind (where user agent normally would choose). Orthogonal to permission.

Migration strategy:

1. Browsers implement pickers for `"user-chooses"` where agent chooses today.
2. Allow sites time to replace in-content pickers in their ⚙️ panel with browser pickers.
3. Remove all labels from `enumerateDevices()`. Deprecate device.getCapabilities()
4. (Maybe) flip default

Criticism / feature (for users w/multiple cams/mics): Flipping default would mean they see a picker even initially, instead of the browser picking the OS default device for them. On sites wo/device selection, they'd be prompted every time (improvement over wrong device).

In-Browser Camera/Mic Picker Model (goals)

2023: No more labels!

Label-less
Device picker-style API

TAG/PING-definition
picker-style API

LEAST POWERFUL

MIGRATION PATH

getUserMedia()++
enumerateDevices()--

no "selectCamera()"
no "selectMicrophone()"

<u>During capture:</u>
All deviceIds
Select capabilities

<u>During capture:</u>
Select deviceIds
Select capabilities

Someday implemented
in all browsers

No consensus on cam/mic
Wouldn't get rid of
getUserMedia or labels

May revisit

Next steps on browser picker

Implementations

- Firefox plans to implement `selectAudioOutput()`
- Hope to gain experience from that (comparable UX challenges)

Other Capture Specifications (Jan-Ivar & Youenn, 30 minutes)

Screen capture

- **Issues**
  - **Issue 60: Define Tab Capture (Harald)**
  - **Is this ready for CR?**

Issue 60: Define Tab Capture (Harald)

- Issue seems editorial
- Spec allows you to capture anything the UA wants to call a “display surface”
- Definition of “browser display surface”:
  - A browser display surface is the rendered form of a single document. This is not strictly limited to HTML [HTML5] documents, though the discussion in this document will address some specific concerns with the capture of HTML.
- This seems to describe tab capture, but what happens if the active document in a tab changes? Text doesn’t seem to say.
- Suggested text:
  - "The UA may choose to display the current document in a browser window, continuing to cast the current document into the same media stream track when the current document changes. This is commonly called tab capture."
- No normative changes needed.

Image Capture

- **Issues**
  - **#256: Clarify meaning of PTZ constraints presence (Jan-Ivar)**

#256 - Clarify meaning of PTZ constraints presence (Jan-Ivar)

Problem 1: (hacky `true` overload) & hand-waving in imageCapture

```
getUserMedia({video: {height: 1080, zoom: true}); // zoom loses (bug)
getUserMedia({video: {height: 1080, zoom: {}});   // ...because it's same as this
getUserMedia({video: {height: 1080});             // ...which is the same as this
```

Proposal: WebIDL: `(double or boolean or ConstrainDoubleOrBoolean) zoom`

```
{zoom: 2}    // aka {zoom: {ideal: 2}}
{zoom: true} // aka {zoom: {ideal: true}}
```

Not complicated, input is either one or the other (existing mediacapture-main algos):

- Fitness if a boolean: (actual == ideal) ? 0 : 1
- Fitness if a number: (actual == ideal) ? 0 : |actual - ideal| / max(|actual|, |ideal|)

#256 - Clarify meaning of PTZ constraints presence (Jan-Ivar)

Problem 2: unspecified whether non-ptz cameras satisfy default values

```
getUserMedia({video: {zoom: 1}); // Regular cameras have 1:1 zoom, so yes? no?
```

Proposal A: They do. This means the following are different

```
getUserMedia({video: {zoom: true}); // Prefers camera with adjustable zoom
getUserMedia({video: {zoom: 2});    // Prefers camera with adjustable zoom
getUserMedia({video: {zoom: 1});    // No camera preference
```

Proposal B: They do not. Specify with prose in mediacapture-main. e.g.

"For all constraints not in the list of inherent constrainable properties, if constraintName is not supported by the device, the fitness distance is 1."

I.e. inherent properties like `{facingMode: "user"}` would be exempt from this rule.

Media Capture from DOM

- **Issues**
  - **#24: Tie capture track lifetime to underlying AudioTrack or VideoTrack, & #85: Define behavior when a cycle is detected (Jan-Ivar)**

#24 / #85 - Tie track lifetime to underlying AudioTrack/VideoTrack

Odd / unimplemented: *"Since a MediaStreamTrack can only end once, a track that is enabled, disabled and re-enabled will be captured as two separate tracks. Similarly, restarting playback after playback ends causes a new set of captured MediaStreamTrack instances to be created. Seeking during playback without changing track selection does not generate events or cause a captured MediaStreamTrack to end."*

Proposal: tie MediaStreamTrack from `.captureStream` to lifetime of AudioTrack/VideoTrack. Have MediaStreamTrack produce nothing (pause) when disabled or restarted.

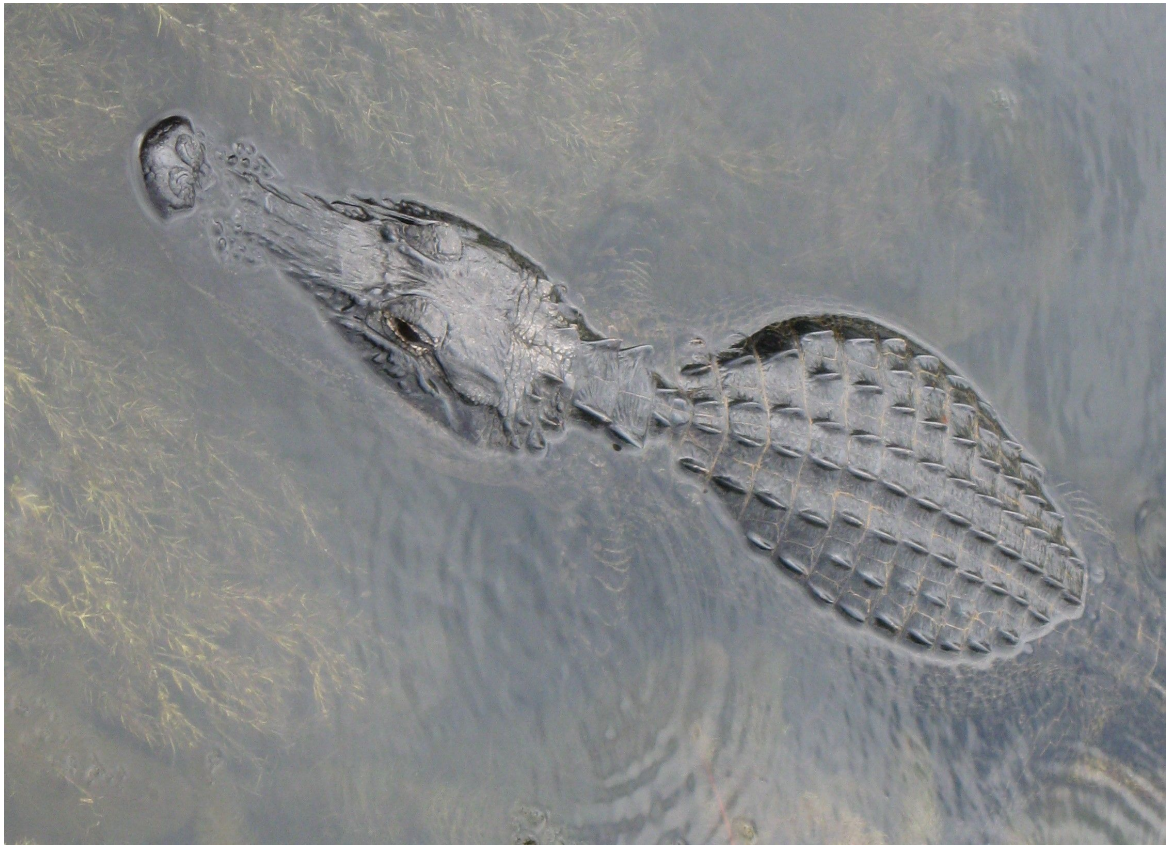
Also fixes cycles (#85), e.g.:

```
element.srcObject = element.captureStream(); // last 20 mins of 2001: Space Odyssey?

element1.srcObject = element2.captureStream();
element2.srcObject = element1.captureStream();
```

...because the `srcObject` load algorithm will remove all selected/enabled tracks, causing the captured MediaStreamTracks to end.

For extra credit



Name that reptile!

Thank you

<u>Special thanks to:</u>

WG Participants, Editors & Chairs

The reptile

Thursday, October 22, 2020

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 Thursday meeting of the W3C WebRTC WG at Virtual TPAC 2020!
- During this meeting, we hope to make progress on new work.

Agenda for Thursday, October 22, 2020

8:00 AM - 8:30 AM Insertable Streams (Raw Media) (Harald)

8:30 AM - 8:50 AM E2E Encryption (Youenn) (https://github.com/w3c/webrtc-insertable-streams/)

8:50 AM - 9:00 AM WebRTC-SVC (Bernard) (https://w3c.github.io/webrtc-svc/)

9:00 AM - 9:15 AM New WebRTC-NV Use Cases (Tim Panton) (https://w3c.github.io/webrtc-nv-use-cases/)

9:15 AM - 9:25 AM GetDisplayMedia and the Same Origin Policy - revisit (Jan-Ivar)

9:25 AM - 9:40 AM GetBrowserContextMedia proposal (eladalon) (https://github.com/w3c/mediacapture-screen-share/pull/148)

9:40 AM - 9:45 AM document.captureStream() MEIG proposal (Jan-Ivar)

[If time allows] Constraint to avoid inefficient pixel formats #739 (Henrik)

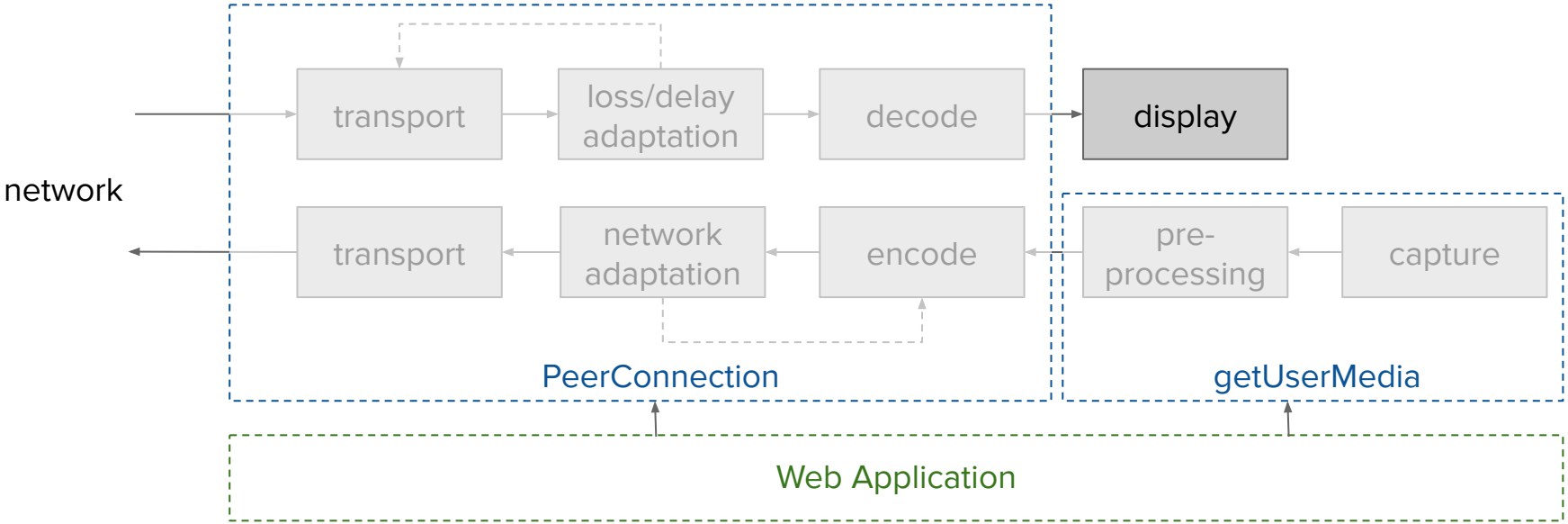
9:45 AM - 10:00 AM Wrapup and Next Steps (Chairs)

Insertable Streams
(Harald, 30 minutes)

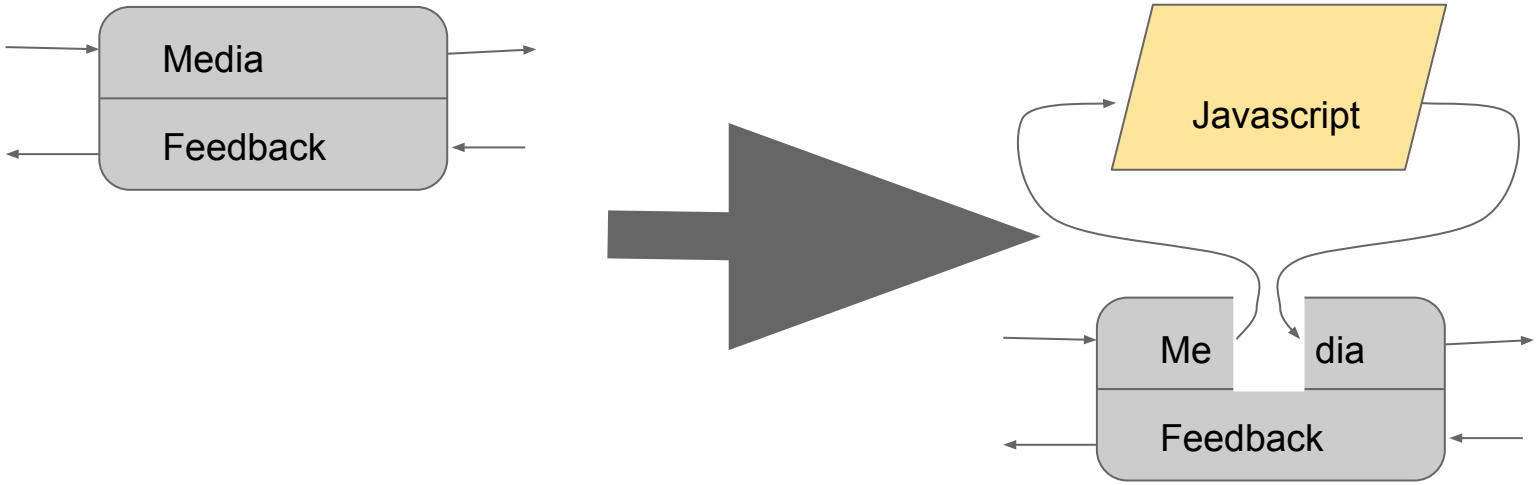
Insertable Streams for Raw Media

- Open up the MediaStreamTrack
- Keep it fast
- Keep it simple

RTC media flow in WebRTC 1.0



Open up the MediaStreamTrack



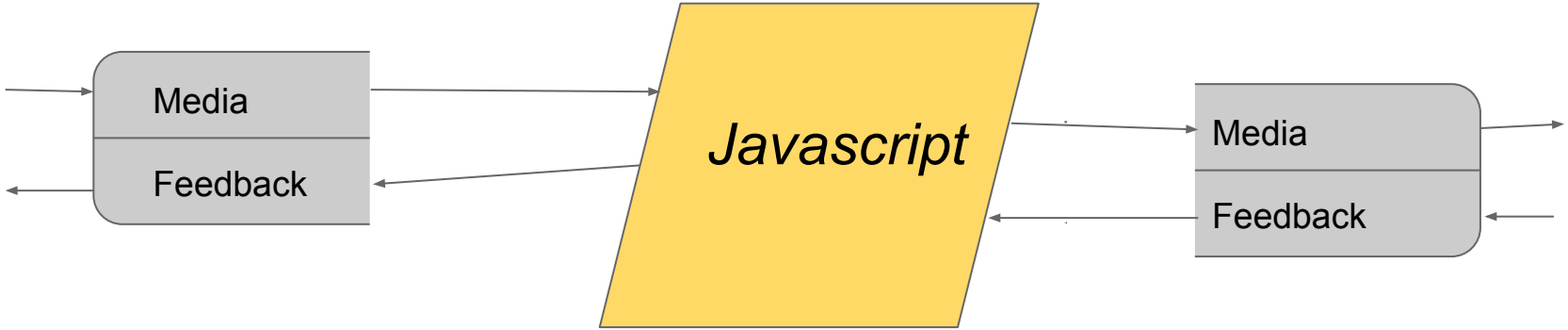
Breakout Box stage 1 -> 2

Stage Two Track Processor

```
function addMoustache(videoFrame) {
  let facePosition = detectFace(videoFrame.data);
  return addMoustache(videoFrame.data, facePosition);
}

processingTrack = new ProcessingMediaStreamTrack(videoTrack);
Transformer = new TransformStream({
   Transform: (videoBuffer) => {
      videoFrame.modifyData(addMoustache(videoFrame));
  }
});
processingTrack.readable
    .pipeThrough(transformer)
    .pipeTo(processingTrack.writable);
```

Break Apart the MediaStreamTrack



Breakout Box stage 3 - allows for generating and consuming tracks directly

WebIDL for two halves of a track

```
interface TrackProcessor : MediaStreamTrack {
    constructor(MediaStreamTrack source);
    attribute ReadableStream readable;  // Stream of VideoFrame
    attribute WritableStream writable;  // Stream of ControlSignal
};

interface TrackGenerator : MediaStreamTrack {
    attribute WritableStream writable;  // Stream of VideoFrame
    attribute ReadableStream readable;  // Stream of ControlSignal
};
```

Why Control Signals?

- Preserve existing behavior
  - A Processor and a Generator coupled should behave ~exactly like a Track
  - Including stop(), active=false, applyConstraints()
- Allow new use cases
  - Inject actions from a worker
  - Act on track events (like “unplug” = stop)

```
dictionary ControlSignal {
  required ControlSignalName name;
  long width;
  long height;
  double frameRate;
  PixelFormat pixelFormat;
};

enum ControlSignalName {
  "stop",
  "mute",
  "unmute",
  "configure",
};
```

Status and Next Steps

- Experimental implementation will be landing in Chrome 88
- Start of specification available
  - https://github.com/alvestrand/mediacapture-insertable-streams

E2E Encryption
(Youenn, 20 minutes)

Native E2E encryption

- E2E Encryption is a desired feature
  - Native applications: Google Duo, FaceTime, Zoom
  - Interest in enabling support for web applications

- Implementation as JS insertable streams?
  - OK for prototyping but need for additional support
  - As per insertable streams explainer:

”This document will also define a mechanism for browsers to provide built-in transformers which do not require JavaScript access to the key or media.”

Native E2E encryption - experiment

- Build upon existing technologies
  - SFrame as underlying format
  - CryptoKey to convey key material
  - Insertable streams to integrate with RTCPeerConnection

- More details available here
  - Similarities with Medooze SFrame

Native E2E encryption - example

Example

```
// Sender
const stream = await getStream();
const pc = new RTCPeerConnection();
const sender = pc.addTrack(stream.getVideoTracks()[0],
stream);

const key = await keyManagement.key();
sender.transform = new SFrameSenderStream();
sender.transform.setEncryptionKey(key);
```

```
// Receiver
const pc = new RTCPeerConnection();
const key = await keyManagement.key();
pc.ontrack = e => {
  e.receiver.transform = new SFrameReceiverStream();
  e.receiver.transform.setEncryptionKey(key);
};
```

WebIDL example

```
// Sender
dictionary SFrameSenderOptions { };
interface SFrameSenderStream : GenericRTCStream {
    constructor(optional SFrameSenderOptions options = { });
    Promise<undefined> setEncryptionKey(CryptoKey key,
        optional unsigned long long keyID);
    Promise<undefined> ratchetEncryptionKey();
    Promise<undefined> setSigningKey(CryptoKey key);
};
```

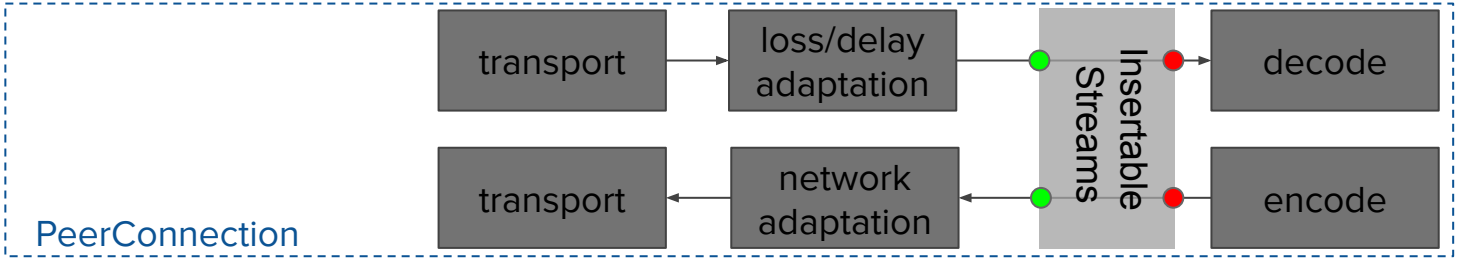
```
// Receiver
dictionary SFrameReceiverOptions { };
interface SFrameReceiverStream : GenericRTCStream {
    constructor(optional SFrameReceiverOptions options = { });
    Promise<undefined> setEncryptionKey(CryptoKey key,
        optional unsigned long long keyID);
    Promise<undefined> ratchetEncryptionKey();
    Promise<undefined> setSigningKey(CryptoKey key);
};
```

Native E2E encryption - conclusiom

- This is feasible
  - And is a timely work item for the WebRTC WG
- Question
  - Should the WG start working on a native E2E transform?
- Proposal
  - Start working on it with the following guidelines
    - No reliance on JS access to keys or media content
    - Support WebRTC mandatory A/V codecs
    - SFrame as the base format

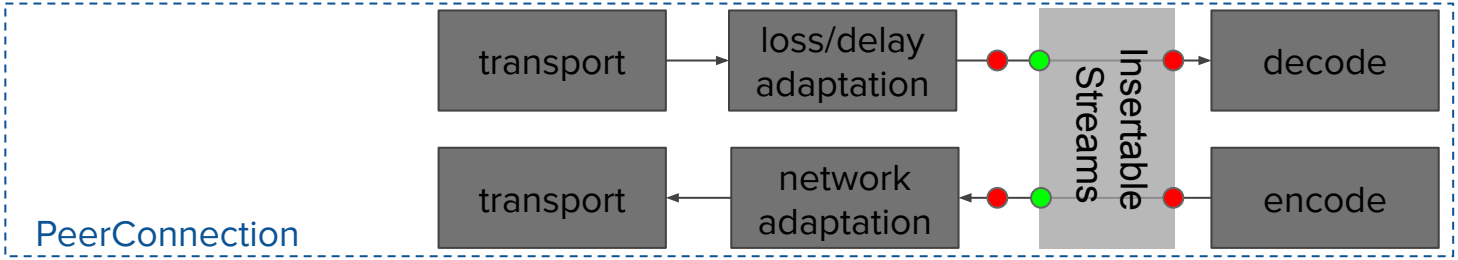
Insertable Streams input/output - 1/2

- ● = valid encoded media content
- ● = valid encoded media content?
- User Agent pipeline might expect ● to be valid media content
  - For packetization/depacketization
- SFUs might expect ● to be valid content
  - Parsing of media to identify useful information (key frames, profile)
- Transforms may not produce valid media content



Insertable Streams input/output - 2/2

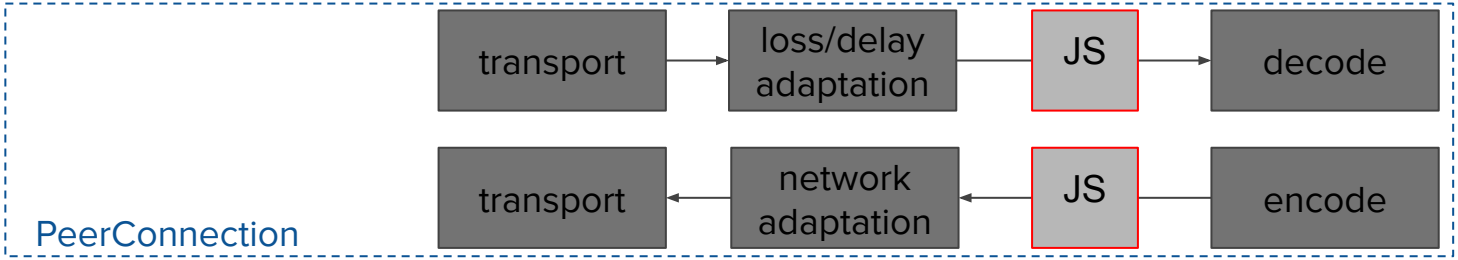
- An optional adaptation transform?
  - Wrap insertable stream output
    - Not needed with generic future packetizer
- Proposal
  - Define this adaptation for H.264 and VP8
    - Within SFrame transform or standalone



Insertable Streams as transforms

- Insertable Streams are transforms by nature
  - For each received chunk, enqueue a modified chunk

- Streams API has support for transforms
  - Transform model is in use
    - Compression Streams, Encoding API



Insertable Transforms - examples

Native Transform

```
// Sender
const key = ...;
sender.transform
  = new SFrameSenderStream();
```

In a Worker

```
// Transform script
function myTransform(input, output)
{
  const key = ...;
  const jsTransform = new JSTransform(key);
  input.pipeThrough(jsTransform)
    .pipeTo(output);
}

// Sender
const worker = ...
sender.transform = new
  RTCTransform(worker, "myTransform");
```

In a Worklet

```
// Transform script
function myTransform(input, output)
{
  const key = ...;
  const jsTransform = new JSTransform(key);
  input.pipeThrough(jsTransform)
    .pipeTo(output);
}

// Sender
const worklet = pc.worklet.addModule("script.js");
sender.transform =
  worklet.createSenderStream("myTransform");
```

Current API

```
// Sender
const key = ...;
const {input, output} = sender.createEncodedStreams()
const transform = new SFrameSenderStream();
input.pipeThrough(transform).pipeTo(output);
```

Insertable Transforms - observations

- API is ergonomic
  - Intuitive use of transforms
  - Allow mixing of native and JS transforms
- Potential for a good threading model
  - Off-the-main thread by default
  - No need for transferring streams in most cases
    - Implementing stream transferring in an optimal way is not that simple
- API exposure is not huge

Insertable Transforms - work(er/let)

- Early to decide
  - Worker and worklet will probably fulfill use cases
- Some worklet advantages
  - Worklet might be more efficient
    - Synchronous processing, reuse of existing threads
  - Worklet might be less error prone
    - No XHR for instance
- More feedback needed?
  - WebAudio WG
  - Native transform stream implementation experiments

WebRTC-SVC
(Bernard, 10 minutes)

WebRTC-SVC Update

- 4 open issues
  - Issue 42: TAG review
  - Issue 22: PING review
  - Issue 14: Encoding parameters for spatial layers (extension)
  - Issue 4: Layer drop/add (extension)
- Implementation
  - “Intent to Implement” in Chrome
    - Available behind “Experimental Web Platform features” flag.
    - Status

WebRTC-SVC Issues

- Issues
  - **Issue 12: Maintenance of scalabilityMode table and diagrams**
  - **Issue 42: TAG review**
  - **Issue 22: PING review: `getCapabilities` seems to leak hardware capabilities w/o a permission**

Issue 12: Maintenance of scalabilityMode table and diagrams

- WebRTC-SVC a good candidate for a “living specification”, so maintenance a key issue.
- Specification referenced AV1 bitstream specification Section 6.7.5 for scalability mode names and diagrams. Disadvantages have become apparent:
  - WebRTC-SVC applies to multiple codecs, not just AV1.
  - AV1 limitations resulted in unnatural names (and diagrams) for some modes.
  - AV1 diagrams can be improved upon (e.g. spatial ID not clear).
- Resolution (now in editor’s draft)
  - Section 6: Scalability modes table
    - Several scalability modes (e.g. KEY and KEY_SHIFT modes) renamed.
    - Links provided to corresponding AV1 scalability mode names.
  - Section 9 (normative) added with scalability mode dependency diagrams in SVG format.
  - Section 6.1 revised to require submission of dependency diagrams along with mode names.

Issue 42: TAG Review

- Question: Why is `scalabilityMode` a DOMString rather than an enum?
  - Enums offer automatic error checking (if a `scalabilityMode` value is not a legal value).
  - Can do `if (mode in scalabiltyMode)` to check validity.
- Section 5.1 specifies (non-automatic) checks:
  - "The `scalabilityMode` selected *MUST* be one of the scalability modes supported for the codec, as indicated in `RTCRtpCodecCapability.scalabilityModes`"

Issue 42: TAG Review (cont'd)

- `scalabilityMode` table has changed as the specification has developed.
  - "S" modes were removed, then added back.
  - KEY and KEY_SHIFT modes were renamed. Example: "L4T7_KEY_SHIFT" is now named "L3T3_KEY_SHIFT"
  - Potential for continued mode additions as new codecs are developed (e.g. AV2).
- Would automatic error checking add value?
  - Non-automatic checks ensure that including a non-supported `scalabilityMode` in `setParameters()` or `addTransceiver()` will result in an error.
  - Automatic checks would enable non-interoperable behavior: one browser could treat L3T3_KEY_SHIFT as an illegal value while another (up-to-date) browser treats it as a legal value, while neither supports that mode.
- Proposal: leave `scalabilityMode` as a DOMString.

Issue 22: `getCapabilities` seems to leak hardware capabilities w/o a permission (WebRTC-SVC specific)

- WebRTC-SVC specification extends the **`RTCRtpCodecCapabilities`** dictionary by adding **`sequence<DOMString> scalabilityModes`**.
- What is the additional fingerprinting surface?
  - WebRTC-SVC currently only supported on Chromium-based browsers (Chrome/Edge/Brave/Opera). So some info provided on the user-agent.
  - SVC encoding currently only implemented in SW, but could change with future HW. SVC decoding capabilities only advertised for SFUs, not browsers. So currently, no SVC-specific HW leakage.
- Issue 49 filed in WebRTC-Extensions.
  - Asserts that existing sync getCapabilities() method is unusable for exposing hardware capabilities.
  - async method proposed to enable querying hw capabilities (and potentially support a permission prompt).
- Proposed resolution.
  - Update the Security/Privacy section to include the above analysis.
  - Address generic getCapabilites() issues in **Issue 2460** and Issue 49

#2460/#22 - getCapabilities leaks hardware capabil w/o permission

A site can learn about the visitor's underlying hardware capabilities w/o a permission prompt or some other positive, affirmative action by the visitor.

Most of the same information is available in the SDP offer from `pc.createOffer()` which inherently needs to be signaled by JS to form a peer-to-peer connection, as described in JSEP (IETF):

4.1.6. createOffer

The createOffer method generates a blob of SDP that contains a [RFC3264] offer with the supported configurations for the session, including descriptions of the media added to this PeerConnection, the codec/RTP/RTCP options supported by this implementation, and any candidates that have been gathered by the ICE agent. An options parameter may be supplied to provide additional control over the generated offer. This options parameter allows an application to trigger an ICE restart, for the purpose of reestablishing connectivity.

In the initial offer, the generated SDP will contain all desired functionality for the session (functionality that is supported but not desired by default may be omitted); for each SDP line, the generation of the SDP will follow the process defined for generating an initial offer from the document that specifies the given SDP line. The exact handling of initial offer generation is detailed in Section 5.2.1 below.

Use cases:

- Data channels
- Receive media
- Send media other than cam/mic/screen, e.g. canvas/elem.captureStream()

#2460/#22 - getCapabilities leaks hardware capabil w/o permission

`getCapabilities`: These capabilities provide generally persistent cross-origin information on the device and thus increases the fingerprinting surface of the application. In privacy-sensitive contexts, browsers can consider mitigations such as reporting only a common subset of the capabilities.

`createOffer` says: The process of generating an SDP exposes a subset of the media capabilities of the underlying system, which provides generally persistent cross-origin information on the device. It thus increases the fingerprinting surface of the application. In privacy-sensitive contexts, browsers can consider mitigations such as generating SDP matching only a common subset of the capabilities.

Conclusion from Graphics Hardware Fingerprinting document linked in issue:

- "Information relating to graphics hardware capabilities provided by [WEBRTC], [WebRTC-Stats], [WebRTC-SVC] ... may also be inferred from other sources such as **Web-GPU, Web-GL and performance API**."
- "...graphics hardware fingerprinting concerns are not WebRTC-specific. ...consider adding a permission relating to *"whether the page is permitted to know what graphics hardware the user is running"* (outside WebRTC)"

Proposed resolution is to include a note relating to implementation issues with hardware capabilities.

New WebRTC-NV Use Cases (Tim Panton, 15 minutes)

New WebRTC-NV Use Cases (Tim Panton)

Theme:
Things that sites are already doing with WebRTC that don’t work as well as they could.

- Low latency P2P Broadcast
- IoT devices in the home or factory or hospital
- Decentralized Internet
- Call robustness
- Reduced complexity signalling

Solution:
Additions to WebRTC could help.

Low latency P2P Broadcast

Target : Auctions and live events

Need : lower latency than HLS/DASH

WebRTC needs to replicate existing streaming features (quality management, codecs, DRM, watermarks, subtitles, ad insertion etc)

Bonus points if it works behind NAT (FTTP)

Low latency P2P Broadcast

Requirement: Maximal reuse of existing higher latency streaming assets

Requirement: new stats of what the recipient sees
Requirement: Clarity on the rules on autoplay

(using service workers to provide webRTC streams that look like DASH/HLS to the DRM capable video element ?)

IoT devices in the home or factory or hospital

Target: IoT device with a camera, confidential video/data is accessed by a browser nearby. (e.g. respiration monitor)

Need: Maximize uptime whilst maintaining e2e privacy

External internet connection may fail

Goal is to ensure continued availability

IoT devices in the home or factory or hospital

Assume: Endpoints have previously been connected.

Requirement: reconnect without recourse to non-local servers.

(STUN and O/A should be optional or predictable for reconnections - possible role for service workers.)

Decentralized Internet

Target: Applications like Matrix foundation’s Element P2P E2E distributed web - from behind NAT.

Need: Ability to have ‘home’ for long running datachannel connection - Page lifecycle is unsuitable for services.

See: P2P Matrix: Where we're going we don't need servers! (https://www.youtube.com/watch?v=QHtw92V2KJQ)

Decentralized Internet

Requirement: Ability to use datachannel connections in a service worker or equivalent.

This would allow a page
to issue a fetch() - which could either be resolved by
a) cache
b) remote public server over https
c) P2P service over webRTC data channel

Call robustness

Target: Mobile communication apps

- currently done as native apps wrapped around libwebrtc
- could be PWAs but….

Need: Call robustness in the face of

a) Incoming GSM call

b) user (un)intentionally leaves page

(e.g. co-browsing support calls)

Call robustness

Requirement: API to select audio/video playout to remote user in the case of a GSM call causing a call to be muted

Requirement: API to reconnect to pre-existing call post GSM interruption

Requirement: API to ‘park’ a track into a service worker on page unload from where it can be retrieved by the next page (if it is in the same origin)

Reduced Complexity Signaling

Target: Client->Server webRTC developers

Need: Simpler signalling that doesn’t require buy-in from as many departments.

A lot of use-cases (e.g. broadcast above) O/A could be replaced by minimal ice-lite-candidate/fingerprint info - enough to bring up a dataChannel. Subsequent Media negotiation can be done over that channel, or just deduced from static config.
See also : https://tools.ietf.org/html/draft-murillo-whip-00

Reduced Complexity Signaling

Requirement: Mechanism to start a session with just a URI

webrtc://${upass}:${ufrag}@192.67.4.185:9010/${fp}/matrix

Contains enough info to start a datachannel labeled ‘matrix’

getDisplayMedia and Same Origin Policy (Jan-Ivar, 10 minutes)

Same Origin Policy

“an important concept in the web application security model.” - Wikipedia

“Under the policy, a web browser permits scripts contained in a first web page to access data in a second web page, but only if both web pages have the same origin. … This prevents a malicious script on one page from obtaining access to sensitive data on another web page through that page's Document Object Model.”

Translation: (scripts,) you can load but you can’t see.

Why: *”servers act based on the HTTP cookie information to reveal sensitive information or take state-changing actions. A strict separation between content provided by unrelated sites must be maintained on the client-side to prevent the loss of data confidentiality or integrity.”*

Without the same-origin policy: *“... a user is visiting a banking website and doesn't log out. Then, the user goes to another site that has some malicious JavaScript code running in the background that requests data from the banking site. Because the user is still logged in on the banking site, the malicious code could do anything the user could do on the banking site. For example, it could get a list of the user's last transactions, create a new transaction, etc. This is because the browser can send and receive session cookies to the banking site based on the domain of the banking site.”*

Same Origin Policy

“...applies only to scripts.” (meaning: barrier between scripts & content)

```
image.src = "https://cross-origin.com/image.png";
canvas.getContext("2d").drawImage(image); // displays but becomes “tainted” 🎨
const data = canvas.getContext("2d").getImageData(0, 0, 50, 50)); // SecurityError


video.src = "https://cross-origin.com/video.mp4";
await video.play();                          // plays fine (ads!)
canvas.getContext("2d").drawImage(video);  // displays but becomes “tainted” 🎨
const data = canvas.getContext("2d").getImageData(0, 0, 50, 50)); // SecurityError



// extends to RTCPeerConnection as well

const stream = video.captureStream();
pc1.addTrack(stream.getVideoTracks()[0], stream); // Sends black
```

QUIZ:
What are obvious screen-sharing risks?

A: I may accidentally share something unintended (wrong tab)

B: Non-visible parts may be included when sharing a window

C: Sharing system audio may reveal other apps or sites I'm using

D: Exposure to active attacks on my browsing sessions

getDisplayMedia and Same Origin Policy

QUIZ:
Do I understand / Can I navigate these risks?

A: Yes

B: Maybe / Yes

C: Maybe / Yes

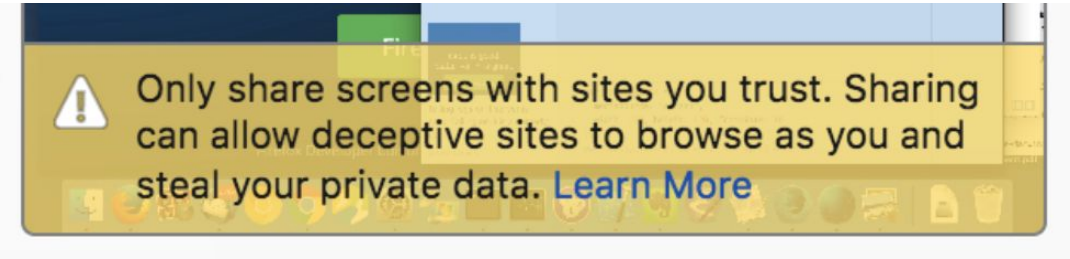
D: No

getDisplayMedia and Same Origin Policy

TL;DR: Not a thing. **`getDisplayMedia` violates the same origin policy.**

Sharing a web surface under attacker control may expose the user to active attacks on cross origins (popping up iframes and embedded media).

Browsers are supposed to require elevated permissions (i.e. differentiate) if users pick a web surface to share:



https://blog.mozilla.org/webrtc/share-browser-windows-entire-screen-sites-trust/

getDisplayMedia and Same Origin Policy

Screen-sharing is an important use case for web to compete with native.

Spec mitigations:

1. Requires elevated permissions to pick a web surface over other choices
2. Sites MUST NOT be able to direct users toward sharing a web surface
3. Full screen is considered a web surface since browsers may be visible
4. Picker every time; no persistent grant permissions allowed

Has undergone TAG security and PING review
https://github.com/w3c/mediacapture-screen-share/blob/gh-pages/questionnaire.md
https://github.com/w3c/mediacapture-screen-share/blob/gh-pages/explainer.md

Capturing a web surface defeats same-origin protections. Capturing a browser window or the desktop when a browser window is visible on it, poses a unique privacy and security risk that is unobvious and significant. If a captured browser window is displaying a page that is under the control of a malicious application, even indirectly, it can allow the malicious application to induce presentation of information that would otherwise be secret from it. For details, see Share browser windows and entire screen only with sites you trust.

GetBrowserContextMedia proposal (eladalon, 15 minutes)

State of the art:

- `getDisplayMedia` allows the app to call upon the user to pick a share-source.
- The source cannot be restricted by the app; i.e. the app cannot specify that it's interested in tab-sharing.
- Some risk of user sharing wrong thing; e.g. whole screen or a different tab.

Proposal:

- Add `getBrowserContextMedia` - allow app to prompt the user for permission to share the current tab.

Benefits:

- Apps can reduce the risk of the user sharing the wrong thing. (See security implications on next slide.)
- The app can reduce remotely-shared information even further, by cropping the video prior to sharing it remotely. Since the app can take for granted that the captured content is of itself, it knows how to crop sensibly. (See use-cases next.)

Use cases:

- Sharing of a document (e.g. Google Slides) to a meeting can be done via a button-click from the page, with a browser prompt to allow/disallow, rather than to choose the current tab.
- Locally recording a [video-conferencing / gaming / etc.] session to a local file. (Also useful for defect-reporting - that captured video can be uploaded.)

Notable considerations:

- The proposal is to model this as closely after getDisplayMedia as possible, in both design as well as implementation, so as to reap the security/privacy benefits of that established API's design and impl.
- Revisiting the decision to not allow `getDisplayMedia` to restrict the share-source is also an option.

Security considerations:

`getDisplayMedia` mandates that the user's selection of capture-source may not be restricted by the app. There is a trade-off here.

- On the one hand, the user might overshare.
- On the other hand, the user cannot be nudged to overshare.

`getBrowserContextMedia` takes a different approach:

- On the one hand, the user cannot accidentally share other tabs, the entire window, or the entire screen.
- On the other hand, the selection is constrained to the most-controlled display-surface - the app's own.
    - Concerning attack vectors include loading of sensitive pages in an <iframe> and extracting data from what is displayed.
        - Banking data would be very concerning, but such sites should typically be secure enough, that the previous risk of the user sharing the wrong tab was greater, than that of the banking site not being ruggedized against such attacks.
        - Displaying links to snoop their visited-status via link-purpling, and similar information, are attacks which could be performed. (However, also with `getDisplayMedia`.)
- We believe that the new trade-off is still desirable:
    - For well-behaved websites, risks to the user are greatly reduced.
    - Even under the worst case scenario, it would only present the same risk, that the existing `getDisplayMedia` presents.
    - Note that the UA is free to warn the user of the risks of sharing the current tab, and place speed-bumps to prevent the user from accepting too readily and unthinkingly.

document.captureStream()

(Jan-Ivar, 5 minutes)

Capture HTML rendering

Today (`getDisplayMedia`)

- *Web-surfaces* may be captured by screen-sharing *only if users pick them*.
- Sharing them carries significant risks not understood by users (malicious sites may do active attacks on WWW's same-origin policy).
- Prevents sites from influencing users to choose these, to deter attacks.
- Behind elevated permission (browsers supposed to warn of risks)

Ironically, sharing native apps is safer.
Unfortunate, since we'd like to promote web over native.

Prohibitive UX flow for "record this meeting" use case or "Present Google Doc"

Capture HTML rendering

Better integration: What if web pages stream themselves into a conference?

The page could use existing tech (`RTCPeerConnection`) to join an ongoing meeting and stream itself there, if it could capture itself.

- The document needs only capture itself.
- To be secure, document must be origin isolated as a matter of policy.
- CORS-only allows opt-in which isn't strong enough, since rendering a document from another origin is different from reading it.
- New policy needed, e.g. `Cross-Origin-Embedder-Policy: disallow`

Capture HTML rendering

More secure, but still needs permission: rendering may contain private info

- link purpling (browser history)
- form autofill (address, credit card info)
- extensions (e.g. LastPass)
- file input element sometimes contains private info

Active attacks could harvest information quickly & covertly (CSS color shading)

These risks are hard to explain to users in a prompt.

Capture HTML rendering

HTML → **Video** a powerful paradigm. Remote browsing; stream web apps

Seems lower-level API than screen-sharing in use cases, behavior, challenges & potential

API suggestions:

- `document.captureStream()` or even
- `canvas.drawImage(document)` if we leave out audio (since we already have `canvas.captureStream()`)

(The latter would put it out of scope for WebRTC)

Might also solve #145 capture screenshot of DOM

(Depending on use cases)

```
canvas.drawImage(document);                    // canvas becomes tainted 🦠

const ctx = canvas.getContext('2d');
const data = ctx.getImageData(0, 0, 50, 50); // SecurityError wo/policy
```

But would still need permission since it exposes same origin rendering with private user information.

```
const bm = await createImageBitmap(canvas, 0, 0, 50, 50); // prompt
```

Today these produce SecurityError on cross-origin content in all browsers
https://jsfiddle.net/jib1/1kz9hfaL/

Avoid expensive pixel formats (Henrik, if time allows)

Issue 739: Add constraint to avoid inefficient pixel formats (Henrik)

Problem:

- For Full HD+ resolutions on external webcams, MJPEG is a commonly supported pixel format because it is compressed (good for USB 2.0 bus capacity).
- The browser has to convert every captured frame to an uncompressed format for rendering, encoding, etc.
  - Converting from MJPEG is expensive.
  - Converting from other (already uncompressed) formats is cheap.
- An application that cares about both quality and performance cannot opt-out from MJPEG.

Issue 739: Add constraint to avoid inefficient pixel formats (Henrik)

How expensive is it?

Measurements on a lightweight “capture@30fps + convert to I420” standalone app (macOS):

Capture Format	Resolution	Normalized CPU Usage [M cycles/s]	Power Consumption [Watt]
NV12 (420v)	640x480 (VGA)	26.51	3.10
...	1280x720 (HD)	28.94	3.23
YUY2 (yuvs)	640x480 (VGA)	20.57	2.98
...	1280x720 (HD)	30.97	3.31
MJPEG (dmb1)	640x480 (VGA)	52.85	4.85
...	1280x720 (HD)	67.99	5.28
...	1920x1080 (Full HD)	102.41	6.27

- YUY2 HD instead of MJPEG Full HD: Reduce CPU load by -70% and save 3 Watts.
- ***Caveats:*** *Mixing measurements from external webcam and built-in webcam. Have not tested in a real browser. You mileage may vary!*

Issue 739: Add constraint to avoid inefficient pixel formats (Henrik)

Proposal:

New boolean constraint: `{avoidInefficientPixelFormats: true}`

Allows the browser to avoid pixel formats that it deems inefficient. Today = MJPEG.

- If a device supports both efficient and inefficient pixel formats, remove all inefficient formats from a device's set of available formats before processing constraints.
  - (Then only lower resolutions and frame rates can be picked.)

- If a device only supports inefficient pixel formats, allow them. It is important to be able to pick the device even if it's MJPEG-only.

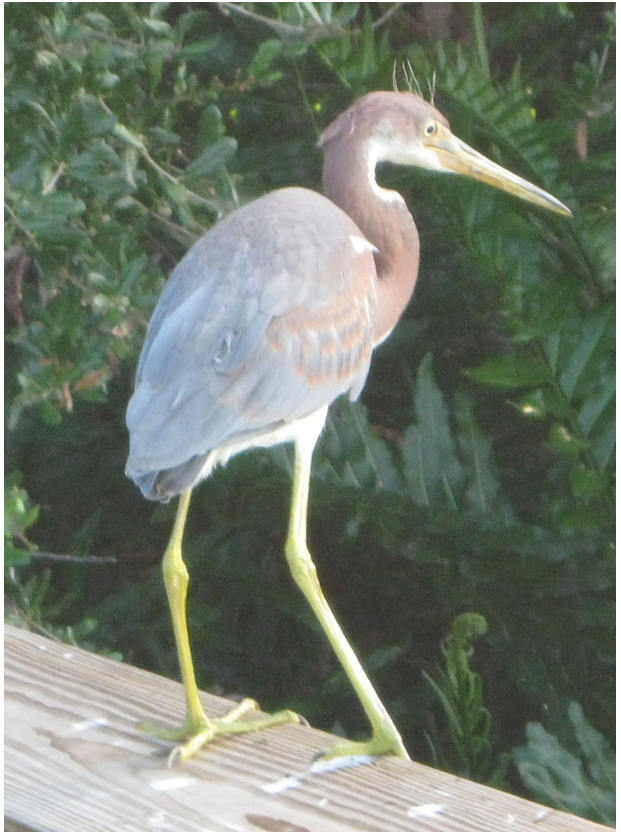
(Details to be fleshed out in a PR, for example, do we need to do a tradeoff between resolution and frame rate?)

Wrapup and Next Steps (Harald, 15 minutes)

Conclusions

<conclusions and next steps go here>

For extra credit



Name that bird!

Thank you

<u>Special thanks to:</u>

WG Participants, Editors & Chairs

The bird